JFreeChart 开发者指南

wizardforcel

Published with GitBook

目錄

JFreeChart 1.0.6 开发者指南

草稿（0.9.0）

译者：乖乖兔

2007-10-25

1 简介

1.1 什么是JFreeChart

1.1.1 概述

JFreeChart是一款免费的java图形开发类库。主要用来在application/ applets/ servlets/ jsp/上生成各种图表。JFreeChart是完全开源，并且严格遵循GNU的通用公共许可证，力保JFreeChart用户对源代码的自由修改与使用。

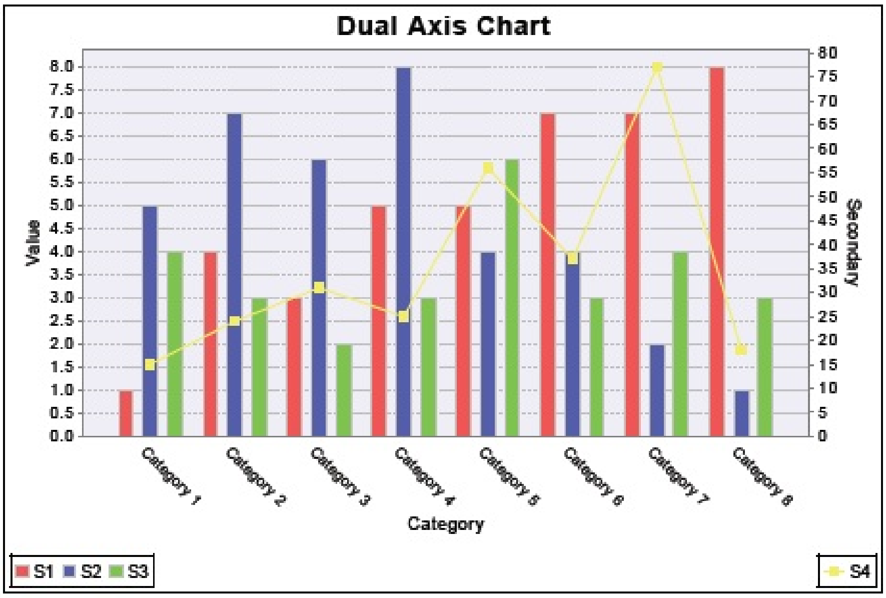


图1.1 一个简单的图表

图1.1就是一个典型的使用JFreeChart创建的图表。在本文后续章节将陆续展示更多的实例。

1.1.2 特征

JFreeChart能产生饼图（pie）、柱状/条形统计图（bar）、折线图（line）、散点图（scatter plots）、时序图（time series）、甘特图（Gantt）、仪表盘图（meter，比如刻度盘、温度计、罗盘等）、混合图、symbol图和风力方向图等。

主要特征如下：

- 定义接口的任何实现通俗易懂
- 易于导出PNG和JPEG图像文件格式（也可以使用java的图像I/O类库生成类库支持的任何格式）。
- 使用Graphics2D工具导出其他格式：
- 使用iText工具导出PDF格式文件
- 使用Batik工具导出SVG格式文件
- 图像工具栏
- 图表支持鼠标事件
- 支持注解。

- 产生HTML图像映射
- 可以工作于application/servlets/jsp/applets等环境。
- 完全开源、严格遵守GNU的通用公共认证协议。

JFreeChart完全由java语言编写，可以运行在java2的任何平台上（JDK1.3.1版本或者更高版本）

1.1.3 下载主页

JFreeChart 可以在下面的链接中找到：

http://www.jfree.org/jfreechart/ 这里我们可以找到JFreechart最新的版本，目前是1.0.6。包括图表实例、下载链接、javadoc文档、讨论社区等。

1.2 使用文档

文档有两个有效的版本：

- 免费版本 可以充JFreeChart主网站上下载免费版本《JFreeChart Installation Guide》，主要讲述内容是：JFreeChart的安装过程和JFreeChart实例的运行。
- 收费版本 需要支付一定费用才能获得《JFreeChart Developer Guide》，主要包括开发指南章节和JFreeChart类参考文档。

1.3 感谢

JFreeChart的代码和思路源于很多人。在这里我将感谢下面帮助JFreeChart成长的人,也许有些人员名字漏掉，望给予指正，在此表示歉意。名单如下：

Richard Atkinson, David Berry, Anthony Boulestreau, Jeremy Bowman, Daniel Bridenbecker, Nicolas Brodu, David Browning, Søren Caspersen, Chuanhao Chiu, Pascal Collet, Martin Cordova, Paolo Cova, Michael Duffy, Jonathan Gabbai, Serge V. Grachov, Hans-Jurgen Greiner, Joao Guilherme Del Valle, Aiman Han, Jon Iles, Wolfgang Irler, Xun Kang, Bill Kelemen, Norbert Kiesel, Gideon Krause, Arnaud Lelievre, David Li, Tin Luu, Craig MacFarlane, Achilleus Mantzios, Thomas Meier, Aaron Metzger, Jim Moore, Jonathan Nash, Barak Naveh, David M. O'Donnell, Krzysztof Paz, Tomer Peretz, Andrzej Porebski, Luke Quinane, Viktor Rajewski, Eduardo Ramalho, Michael Rauch, Cameron Riley, Dan Rivett, Michel Santos, Thierry Saura, Andreas Schneider, Jean-Luc Schwab, Bryan Scott, Roger Studner, Irv Thomae, Eric Thomas, Rich Unger, Daniel van Enckevort, Laurence Vanhelsuw´e, Sylvain Vieujot, JelaiWang, MarkWatson, Alex Weber, Matthew Wright, Christian W. Zuckschwerdt, Hari and Sam (oldman).

1.4 建议

如果您对本文档有任何的建议或想法，请发送：david.gilbert@object-refinery.com。

2 图表实例

2.1 介绍

本章节显示了许多使用JFreeChart创建的图表实例。内容特意对JFreeChart产生的图表类型做了概述。运行实例命令如下：

```
java -jar jfreechart-1.0.6-demo.jar
```

如果您购买了《JFreeChart Developer Guide》，可获得该实例的源代码。

2.2 饼图（Pie Charts）

JFreeChart能够创使用符合PieDataset接口标准的数据创建饼图。下图2.1显示了一个简单的饼图。

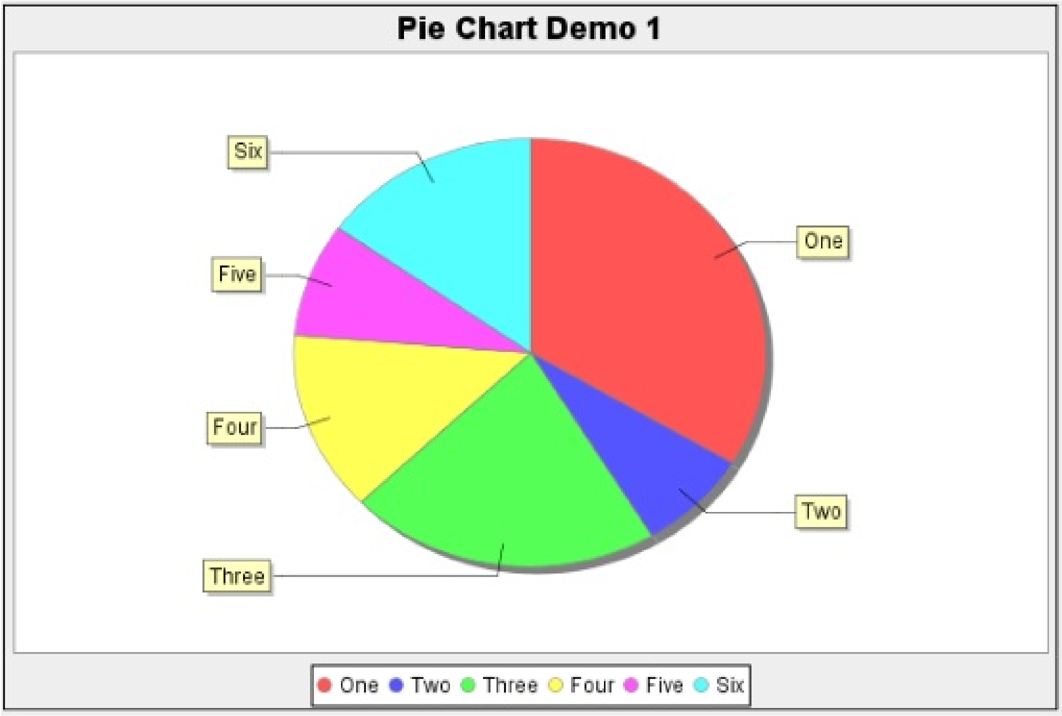


图2.1 一个简单的饼图（参见：PieChartDemo1.java）

其中， 单个的区域也可以被“取出”，如下图2.2所示：

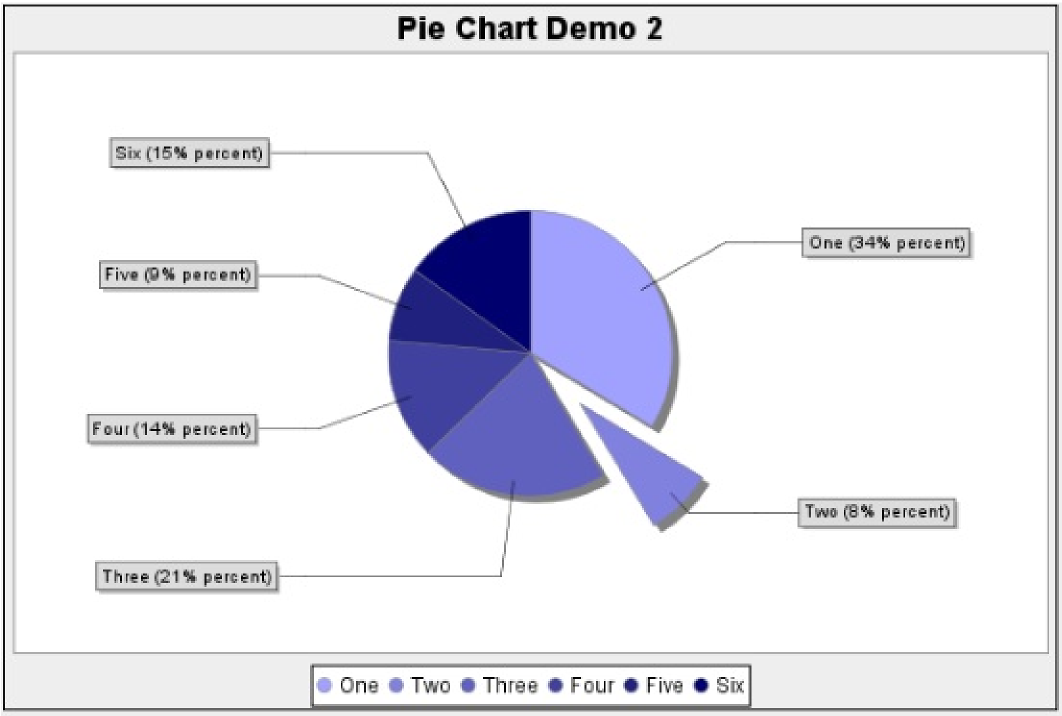


图2.2 一个带有取出“区域”的饼图（参见：PieChartDemo2.java）

我们也可以显示3D效果的饼图，如下图2.3所示：

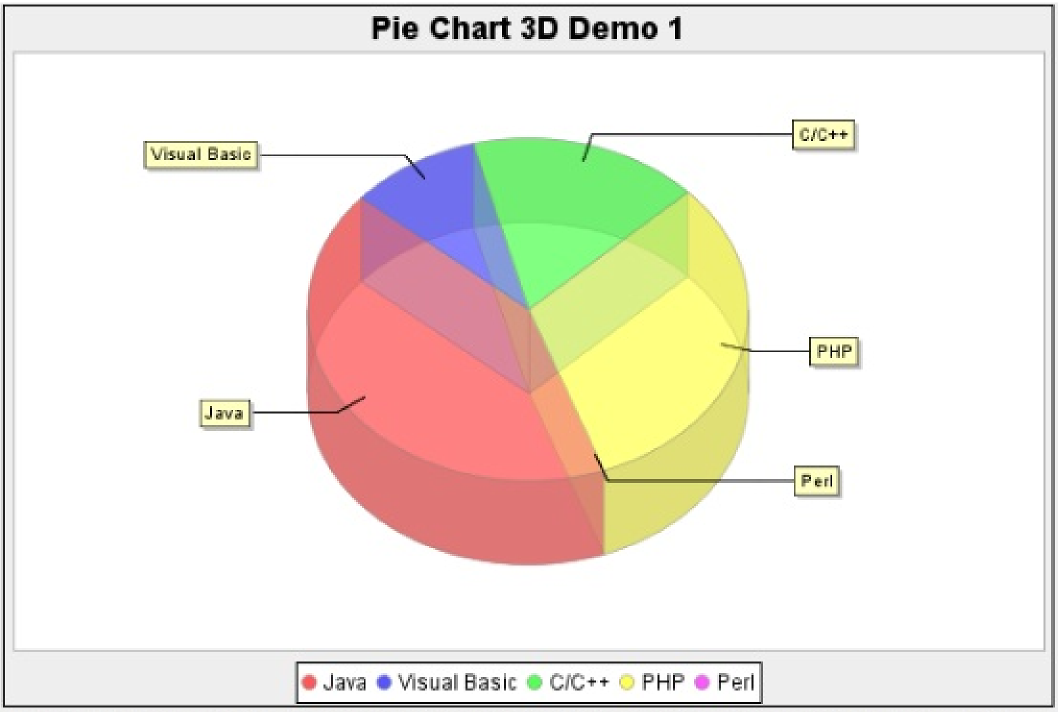


图2.3 3D效果图的图表（参见：PieChart3DDemo1.java）

3D效果的饼图，部分区域不能取出。

2.3 直方条形图（Bar Charts）

JFreeChart可以创建一系列的直方条形图。创建直方条形图的数据必须符合CategoryDataset接口标准。图2.4显示了一个垂直定向的直方条形图。

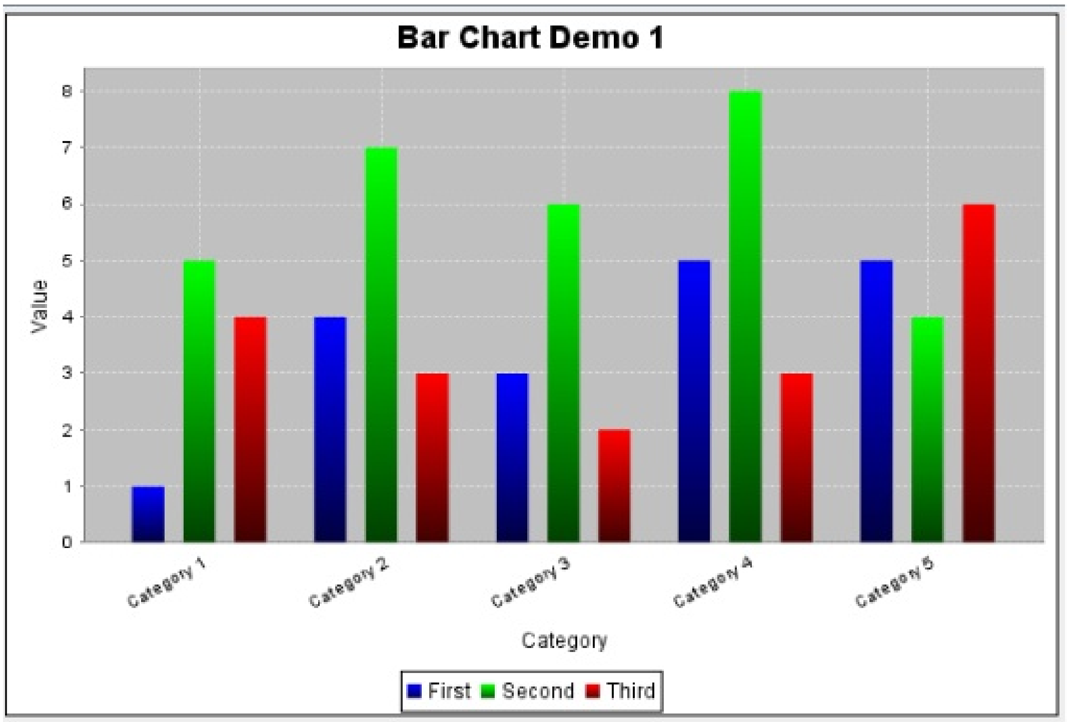
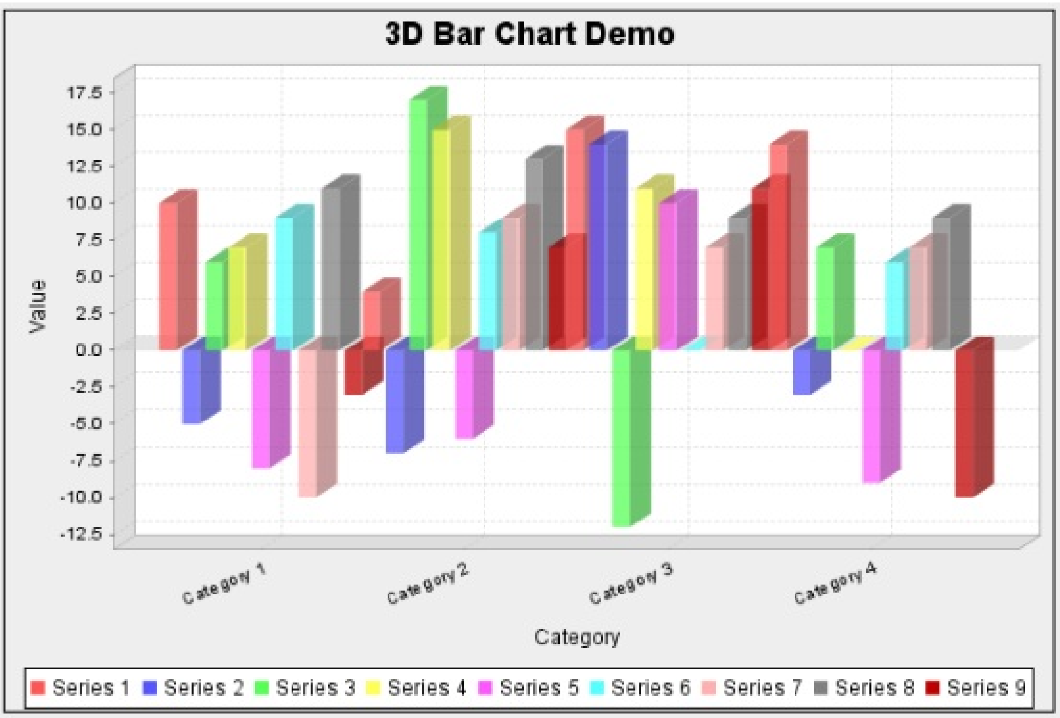


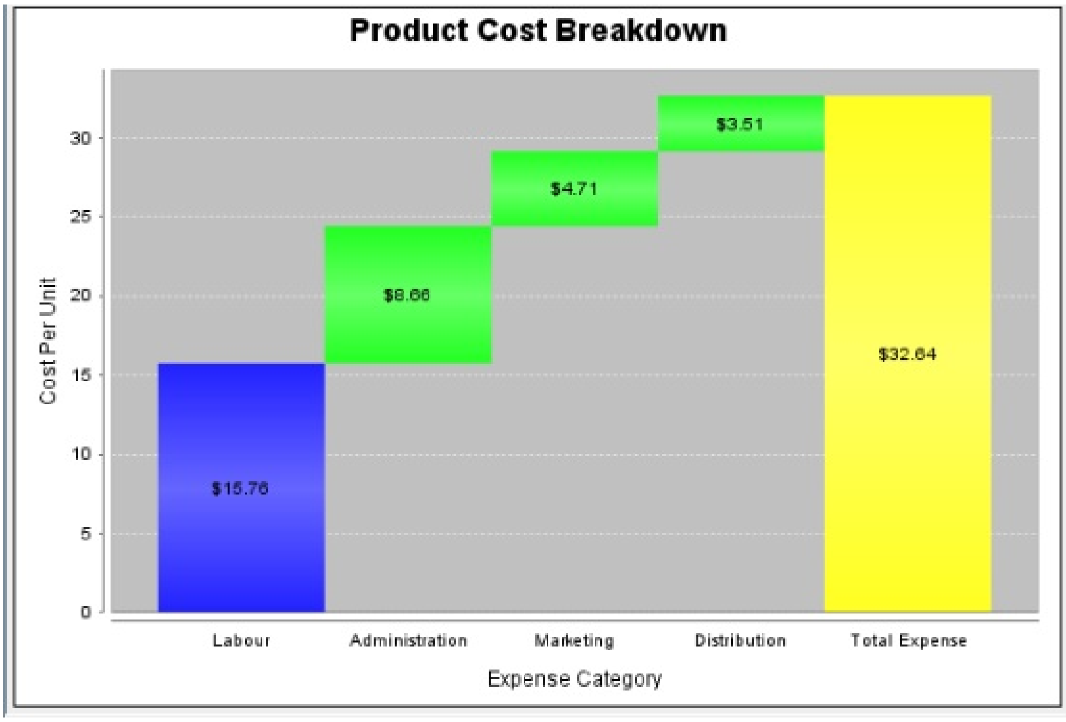
图2.4 一个垂直的直方条形图（参见：BarChartDemo1.java）

直方条形图可以用3D效果显示，如下图2.5所示。



如2.5 3D效果的直方条形图（参见：BarChart3DDemo1.java）

直方条形图的另一种变型，瀑布图表。如下图2.6所示：



如2.6 一个瀑布图表（参见：WaterfallChartDemo1.java）

直方条形图可以从时序数据中产生。如下图2.7所示：

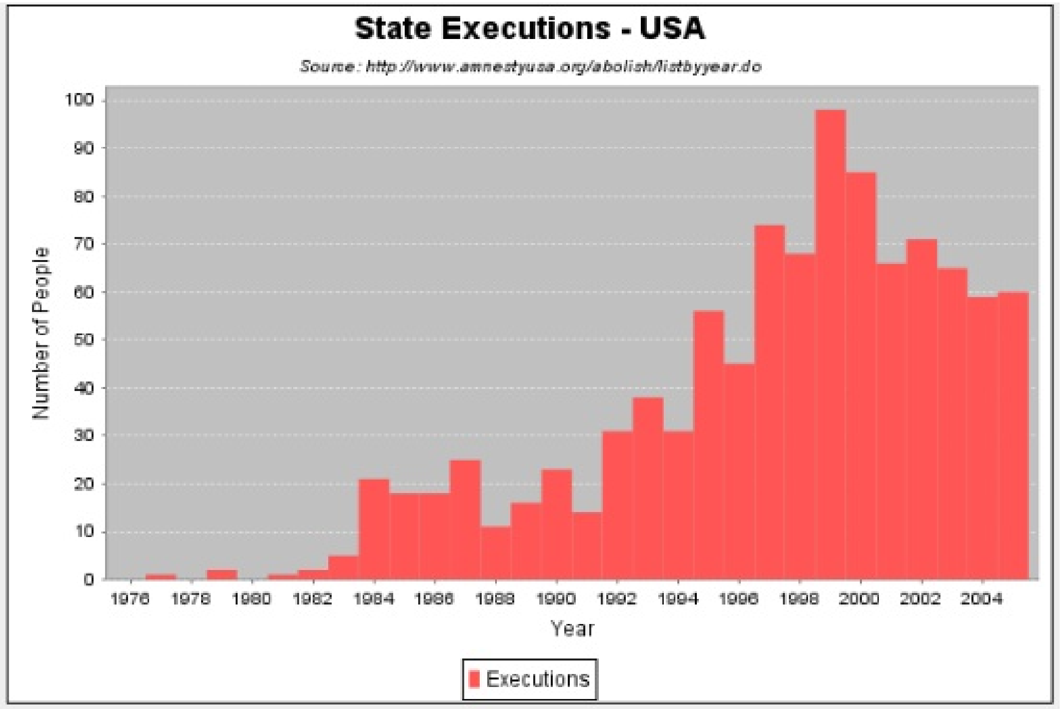


图2.7 一个XY图表（参考 XYBarChartDemo1.java）

2.4 折线图（Line Charts）

折线图可以使用直方条形图的数据对象CategoryDataset产生。如下图2.8：

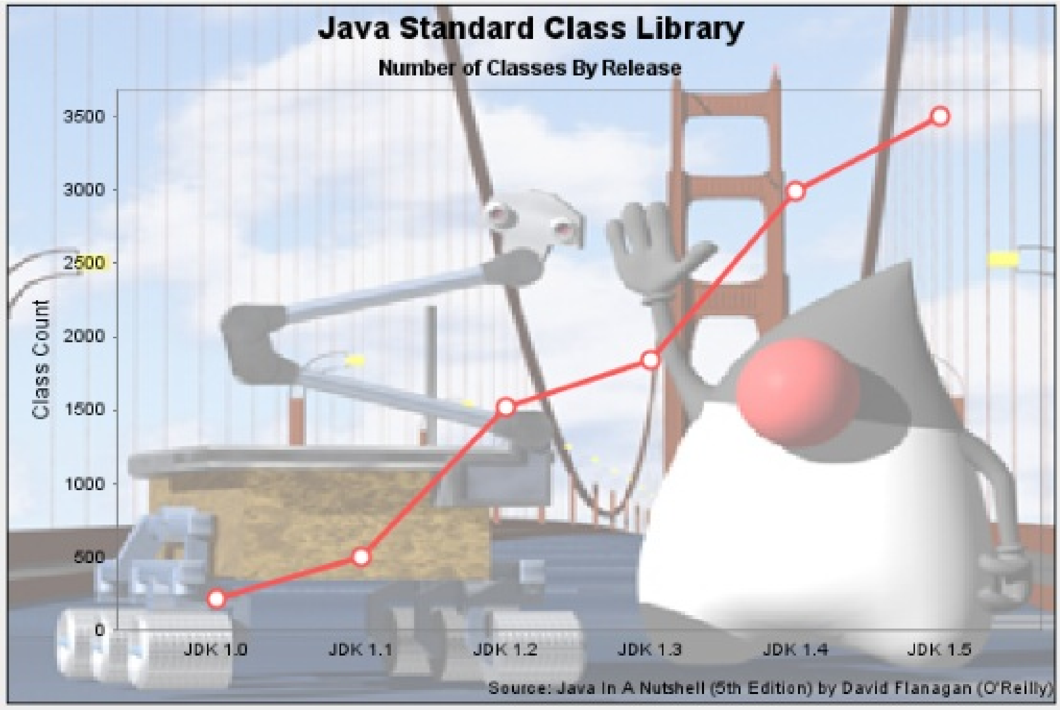


图2.8 一个折线图（参考LineChartDemo1.java）

2.5 XY(散点图)

XYDataset是第三种数据类型，用来产生一系列图表的类型。标准的XY区域有X和Y数轴。默认的，使用相应的数据按照一定比例画出X轴和Y轴。如图2.9所示。

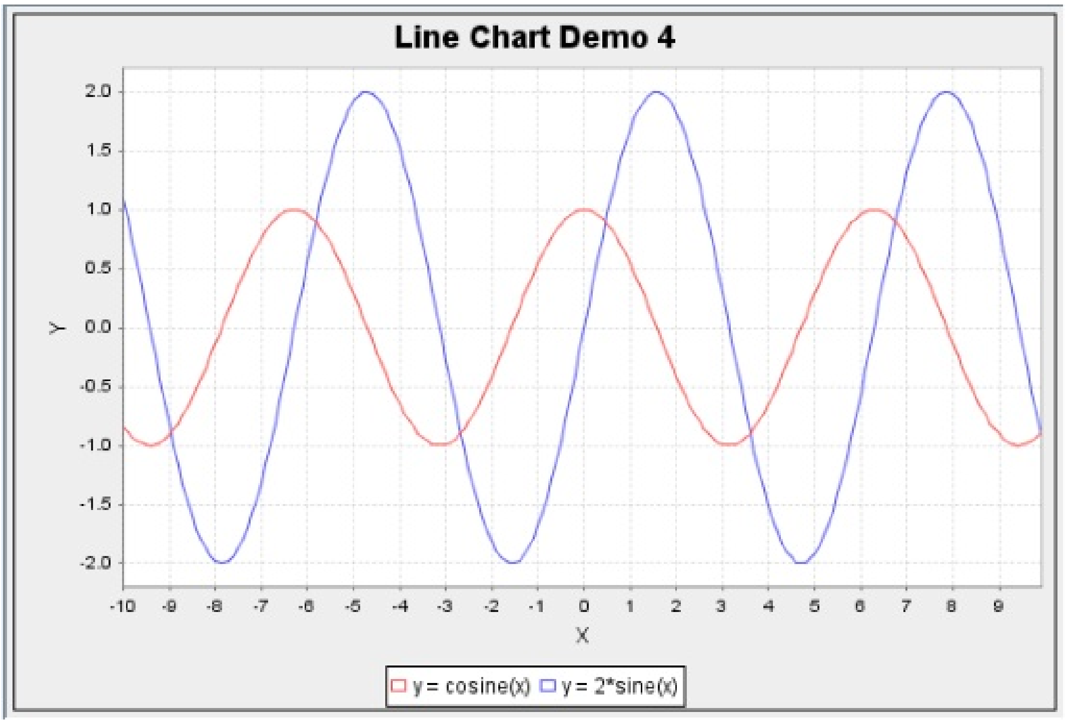


图2.9 折线图（参考：LineChartDemo4.java）

散点图是每一个数据点用一个图形画出来，而不是使用线将点连起来。一个实例如下图2.10所示：

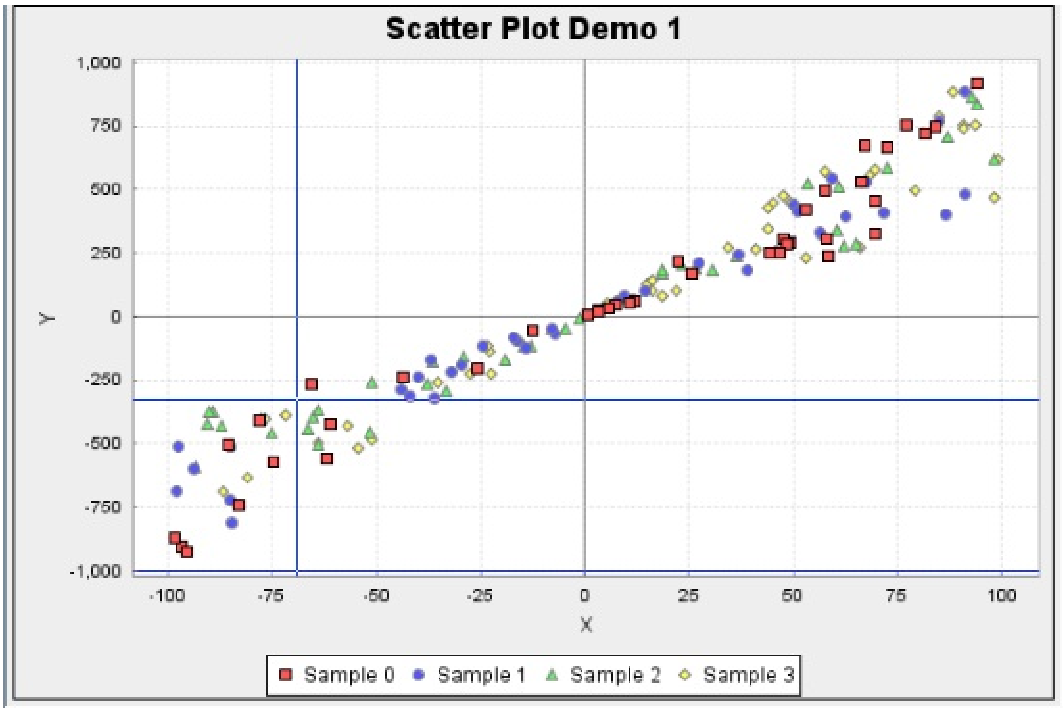


图2.10 散点图（参考：ScatterPlotDemo1.java）

2.6 时序图

JFreeChart支持时间序列图表，时序图包括平均值图、high-low-open-close图和candlestick图，如下图2.11所示：

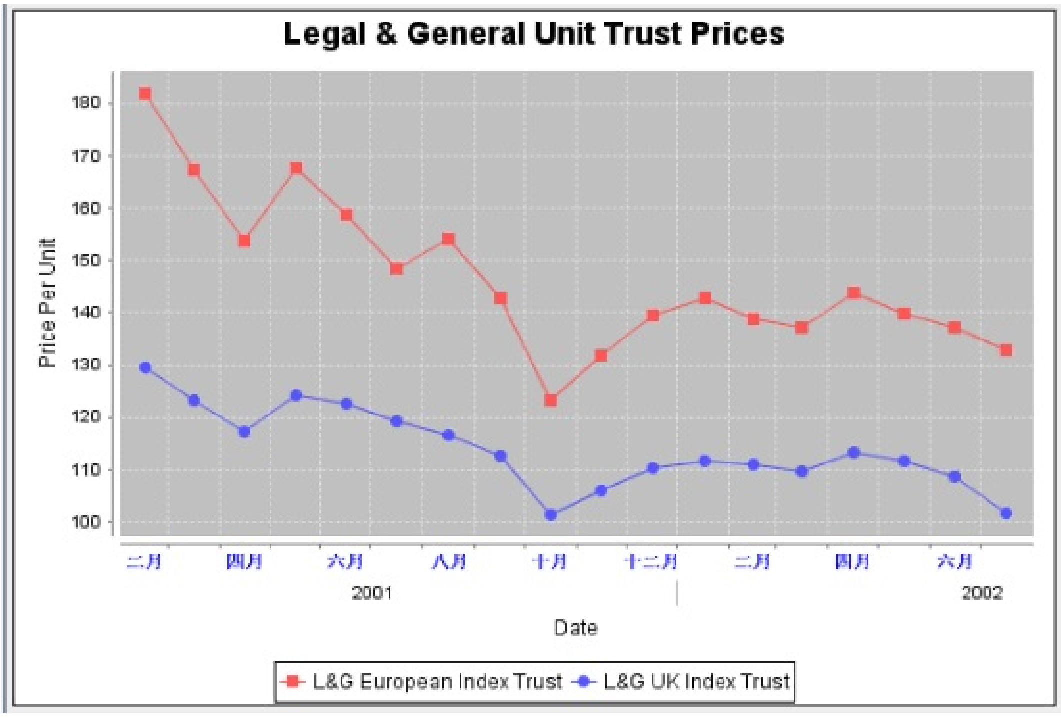


图2.11 序列图（参考：TimeSeriesDemo1.java）

我们可以在时序图上添加一条平均值线——如下图2.12所示：

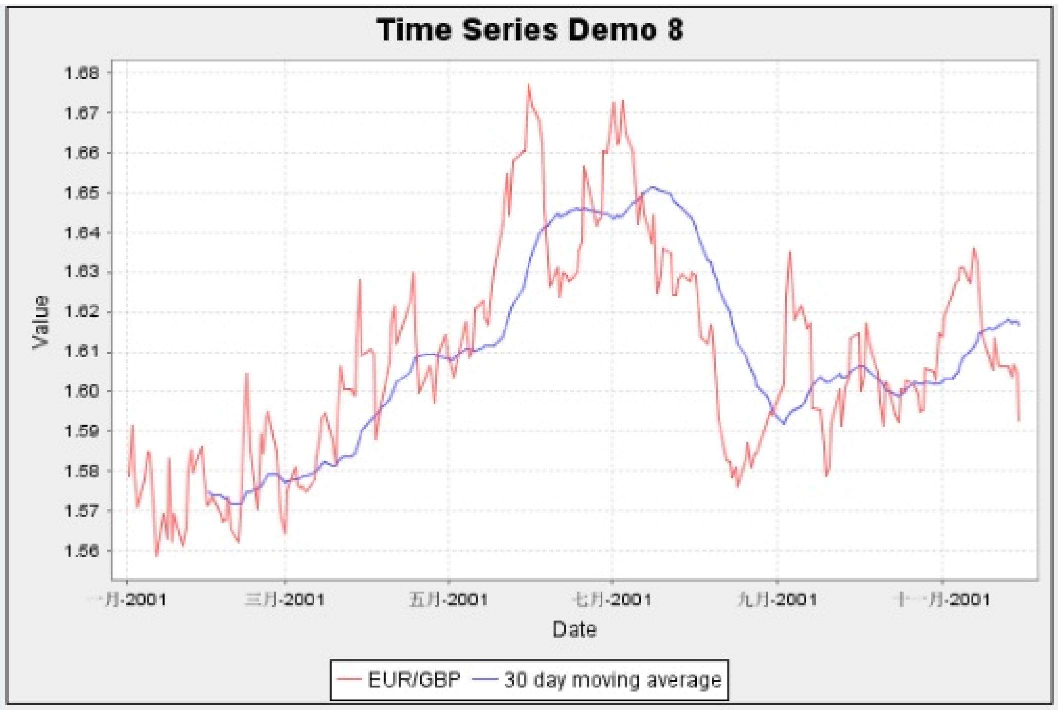


图2.12 带有平均线线的时序图（参考：TimeSeriesDemo8.java）

我们可以使用OHLCDataset（XYDataset的扩展）显示high-low-open-close数据图表。如下图2.13所示：

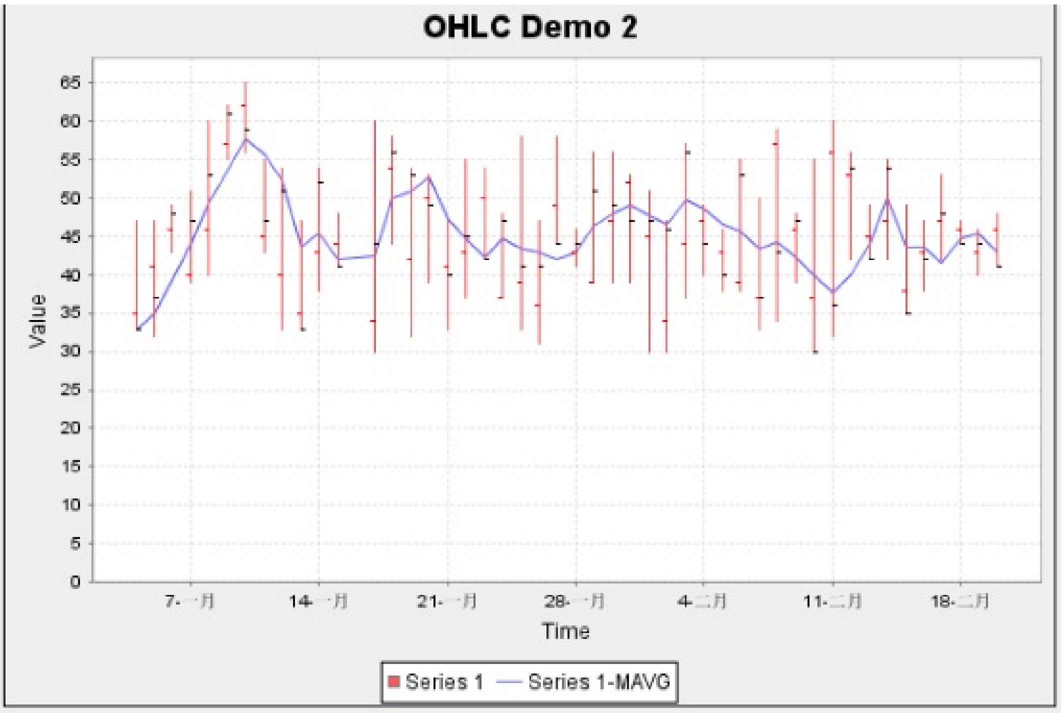


图2.13 high-low-open-close图表（参考：HighLowChartDemo2.java)）

2.7 柱状图

可以使用一个IntervalXYDataset（XYDataset的另一个扩展）数据产生柱状图。如下图2.14所示：

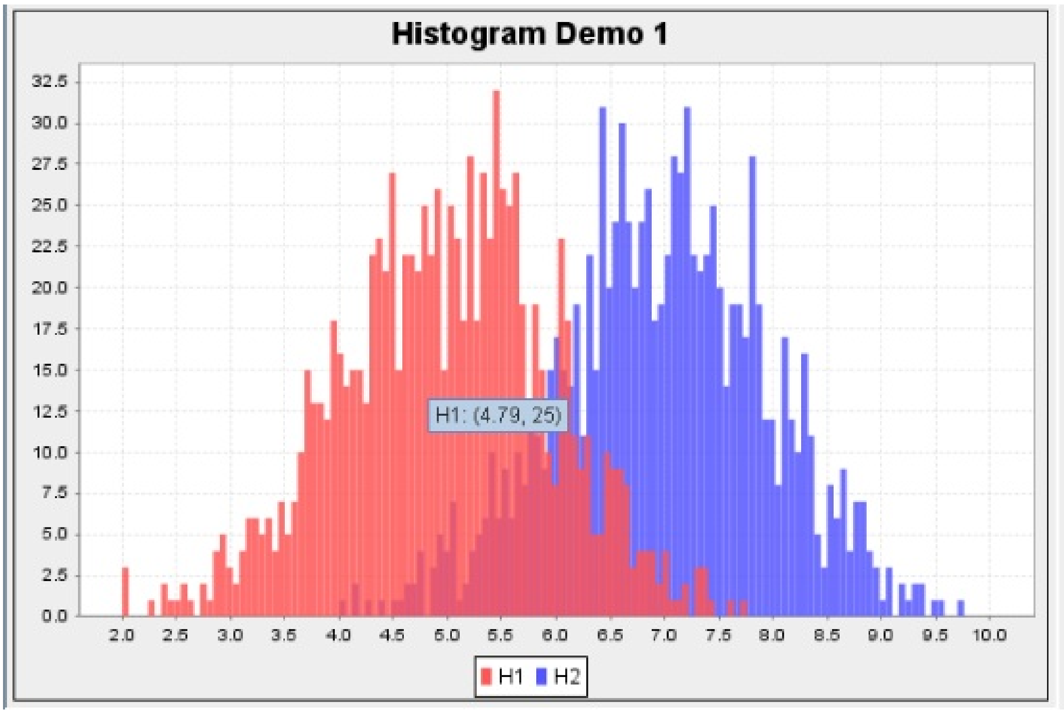


图2.14 柱状图（参考：HistogramDemo1.java）

2.8 面积图

我们可以使用CategoryDataset或者XYDataset产成面积图表。如下图2.15所示：

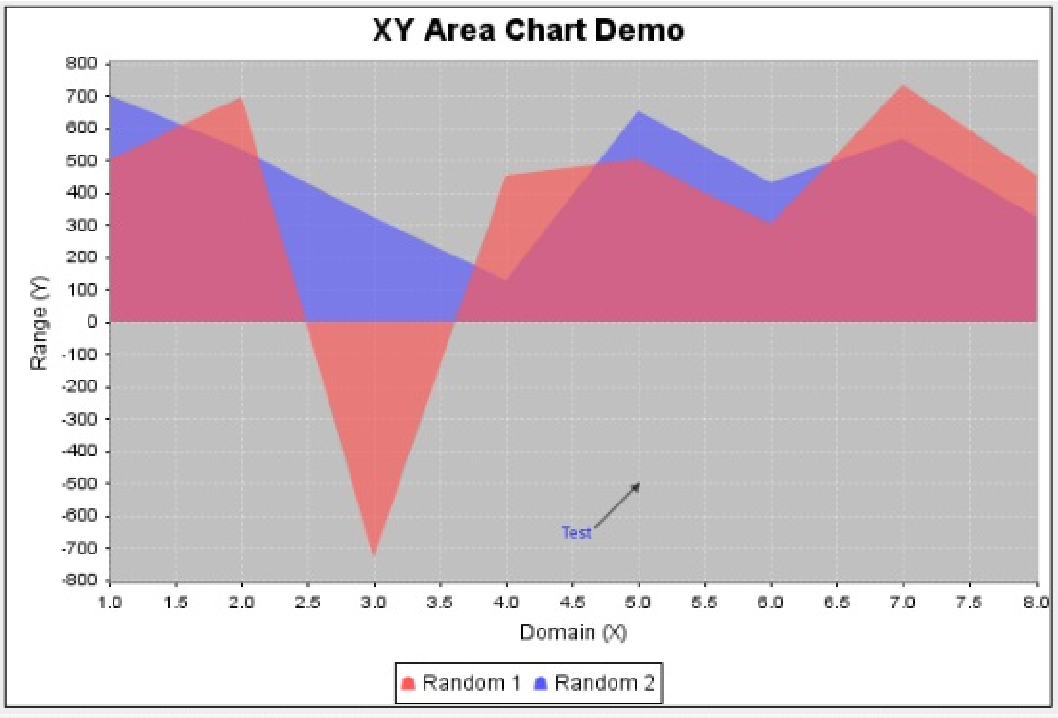


图2.15 面积图（参考：XYAreaChartDemo1.java）

同时，JFreeChart也支持堆栈式面积图表，如下图2.16所示：

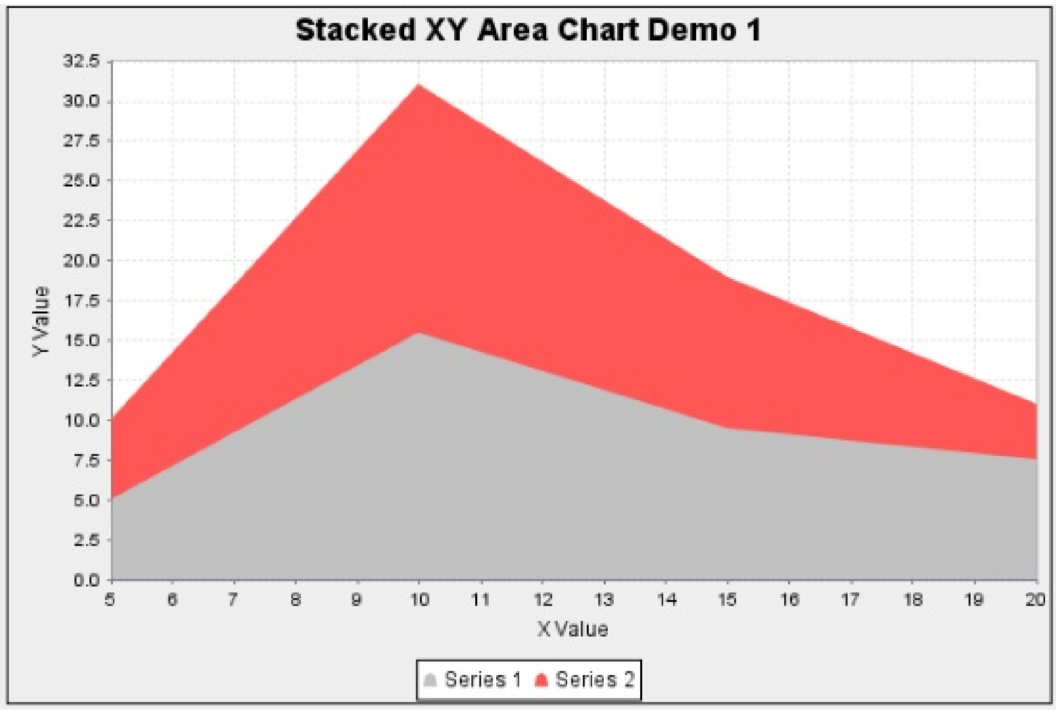


图2.16 堆栈式面积图（参考：StackedXYAreaChartDemo1.java）

2.9 差异图

差异图是显示两个序列之间的不同。如下图2.17所示。

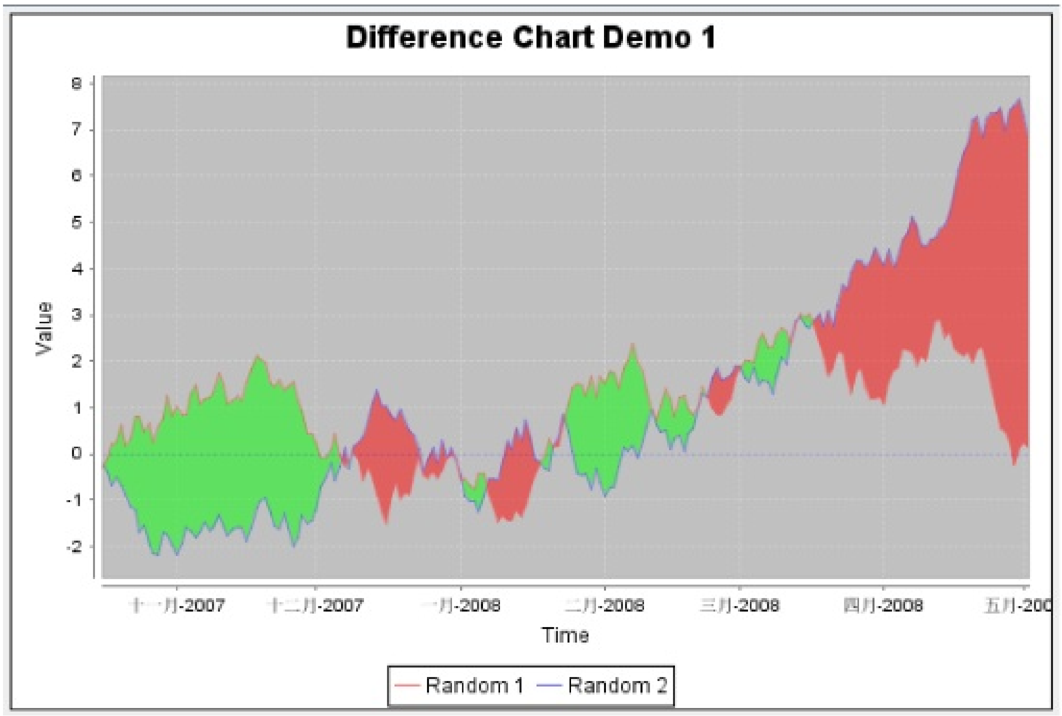


图2.17 差异图（参考：DifferenceChartDemo1.java）

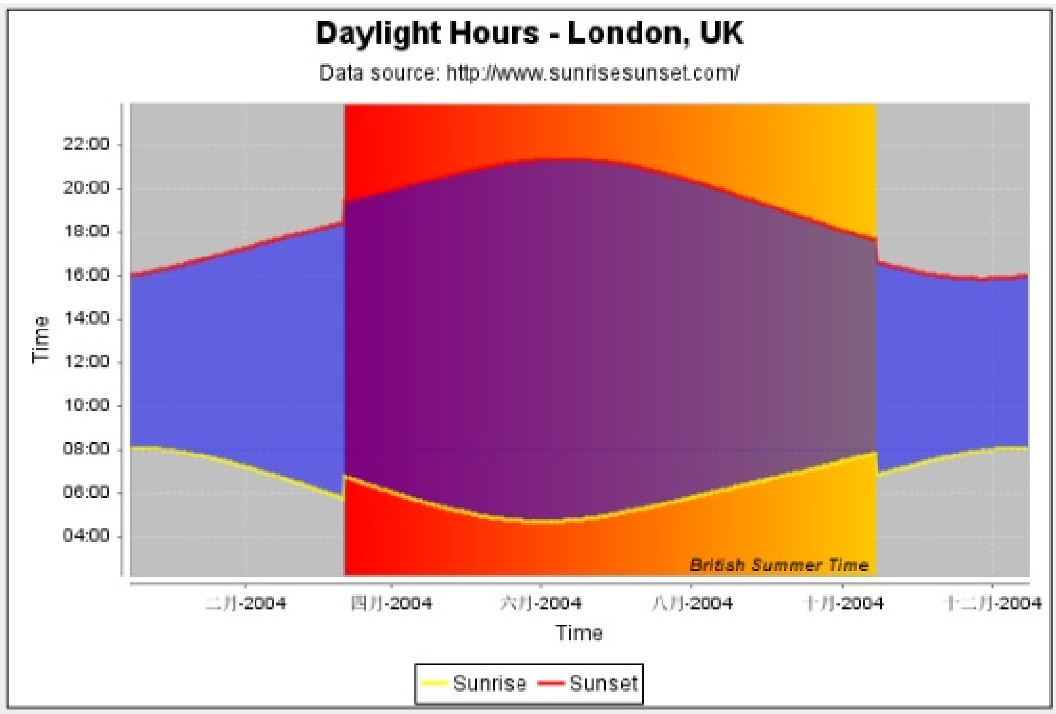


图2.18 差异图（参考：DifferenceChartDemo2.java）

2.10 梯形图

梯形图使用一系列的“梯形”来显示数据数值。——如下图2.19所示：

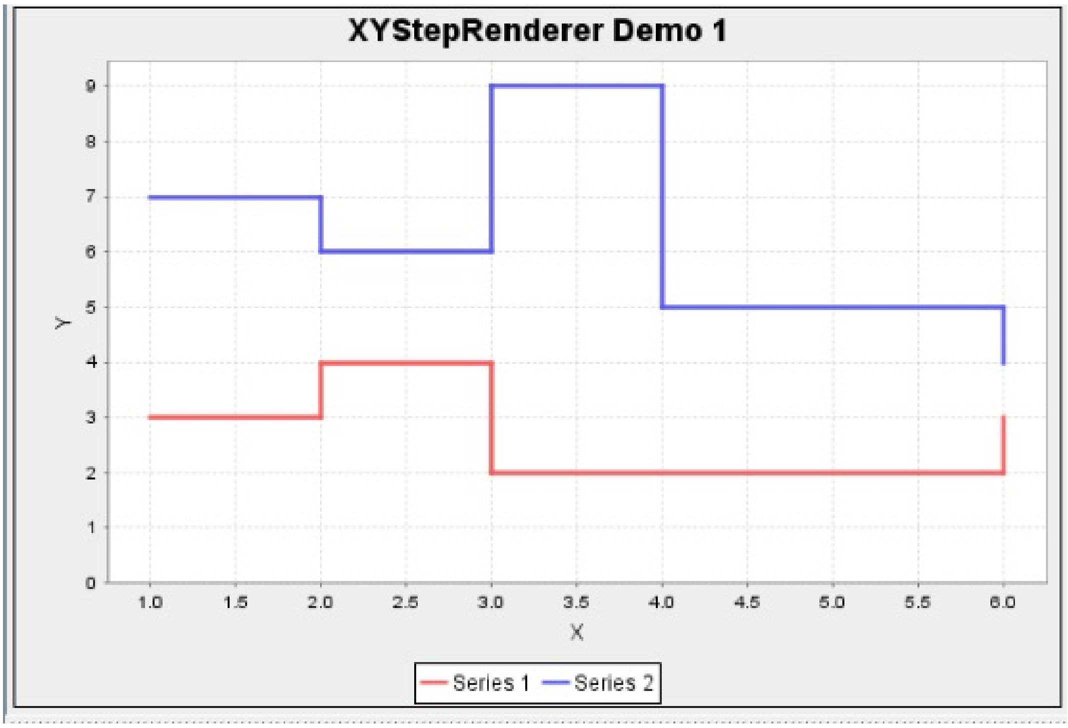


图2.19 梯形图（参考：XYStepRendererDemo1.java）

梯形图数据使用XYDataset数据对象。

2.11 甘特图

我们可以使用IntervalCategoryDataset数据集类产生甘特图。如图2.20所示

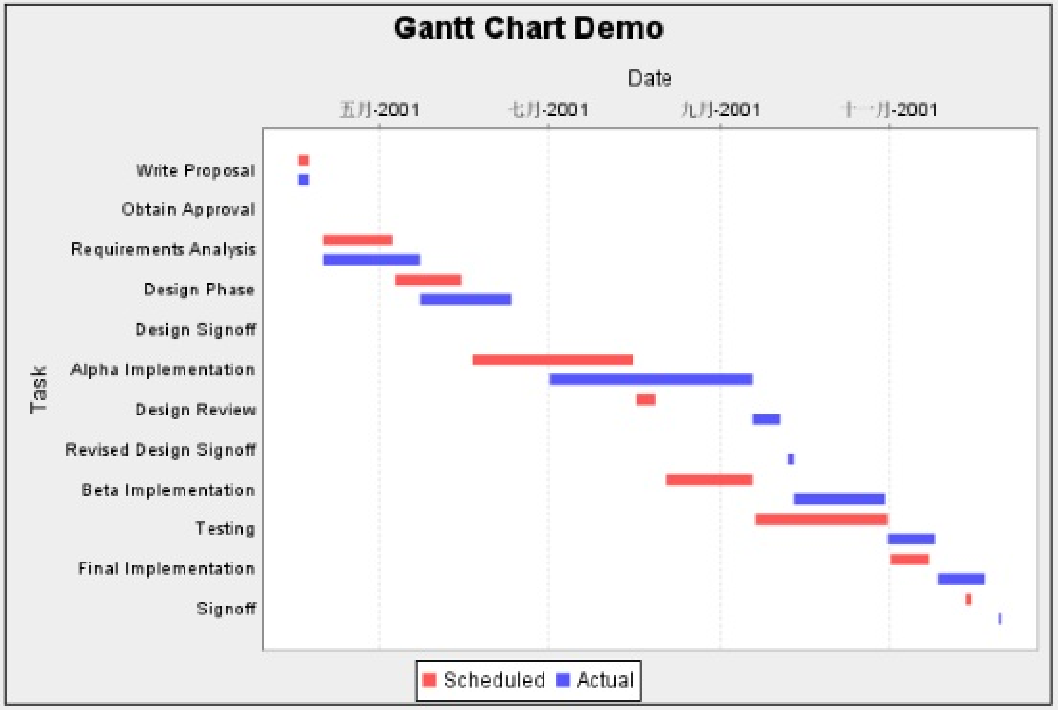


图2.20 甘特图（参考：GanttChartDemo1.java）

此外，甘特图可以具有子任务和进度显示器。如下图2.21所示

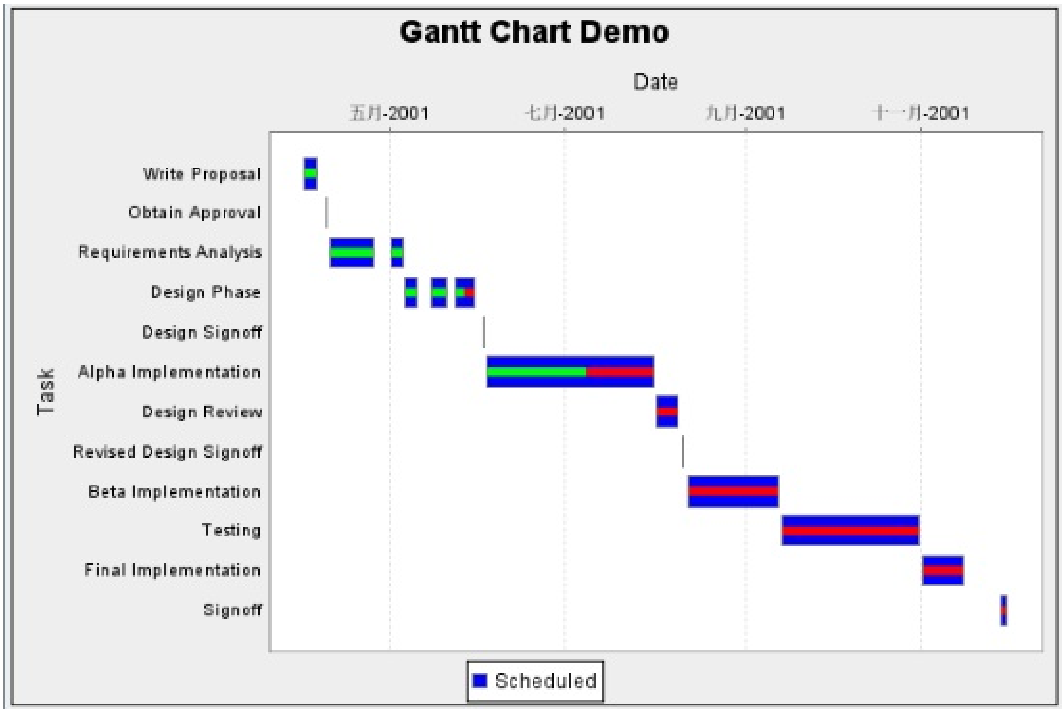


图2.21 带有进度显示的甘特图

2.12 多轴图

JFreeChart支持多轴图表。如下图2.22显示了一个价格－数量的图表。

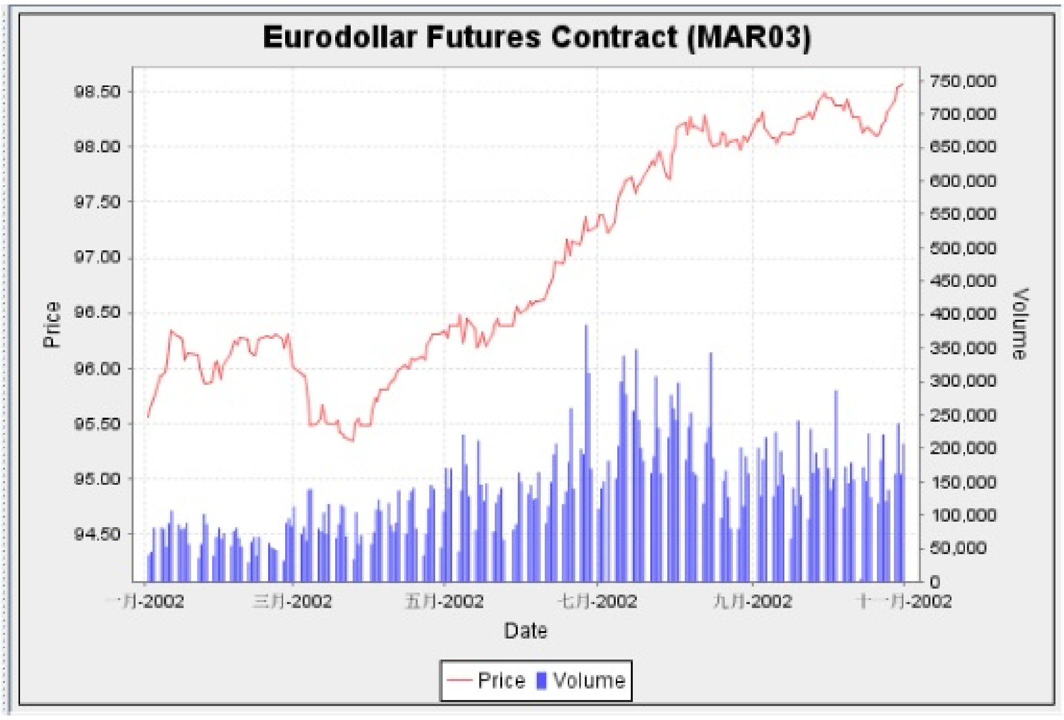


图2.22 价格－数量图表（参考：PriceVolumeDemo1.java）

CategoryPlot和XYPot支持多轴特征。图2.23显示了一个具有四个数轴的图表。

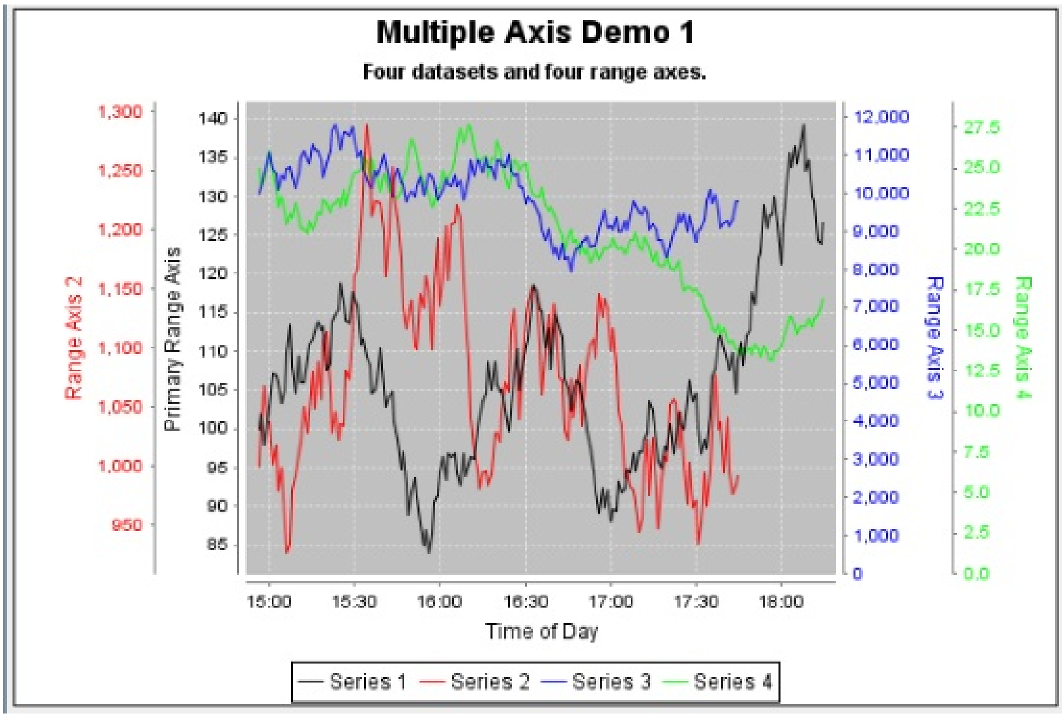


图2.23 多轴图表（参考：MultipleAxisDemo1.java）

2.13 复合/覆盖图

JFreeChart支持复合/覆盖图表。图2.24显示了一个条形图上覆盖了一个折线图。

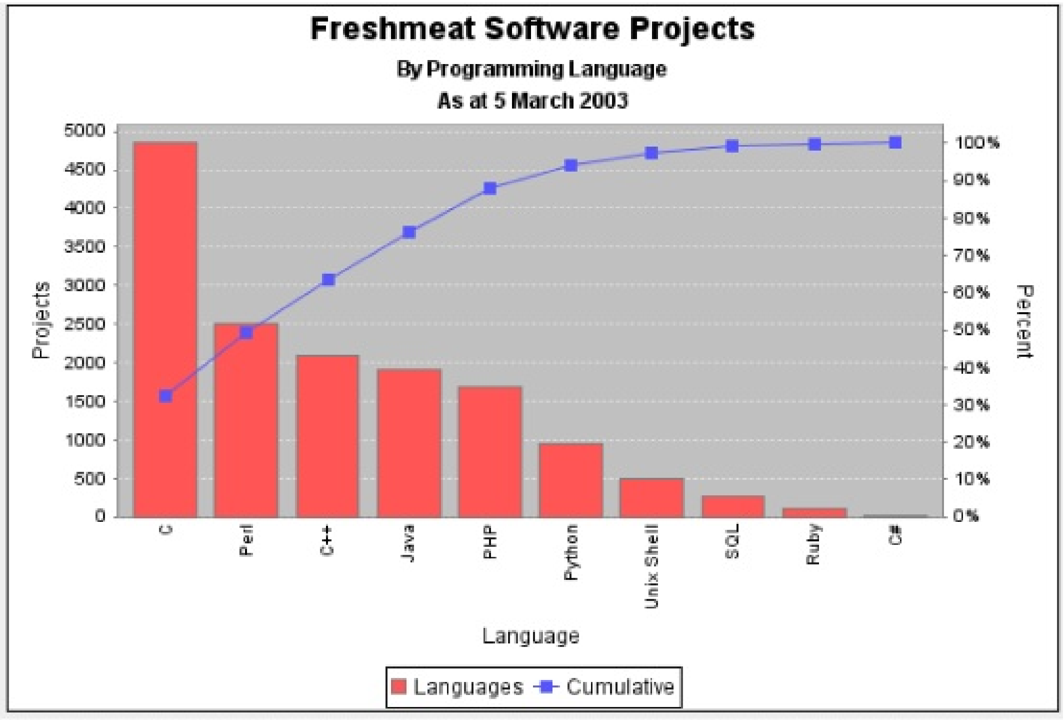


图2.24 覆盖图（参考：ParetoChartDemo1.java）

也有可能使用同一个主轴，组合几种图表。如下图2.25。

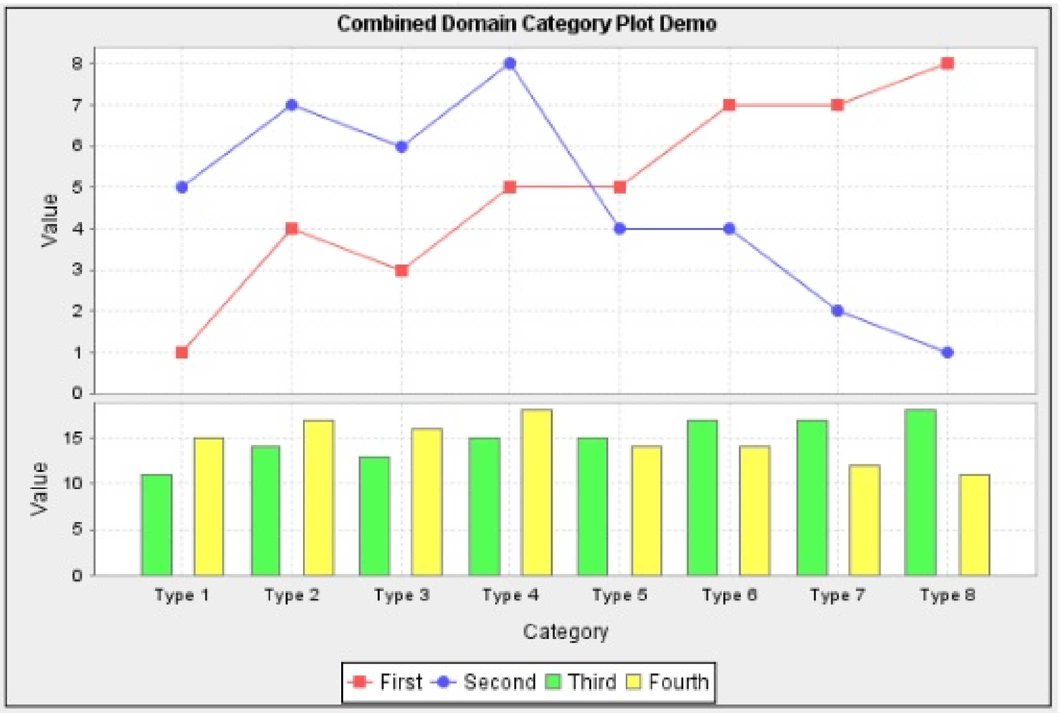


图2.25 带有公共区域的图表（参考：CombinedCategoryPlotDemo1.java）

类似的，JFreeChart可以复合几种图表，共用相同刻度范围的轴。如图2.26所示：

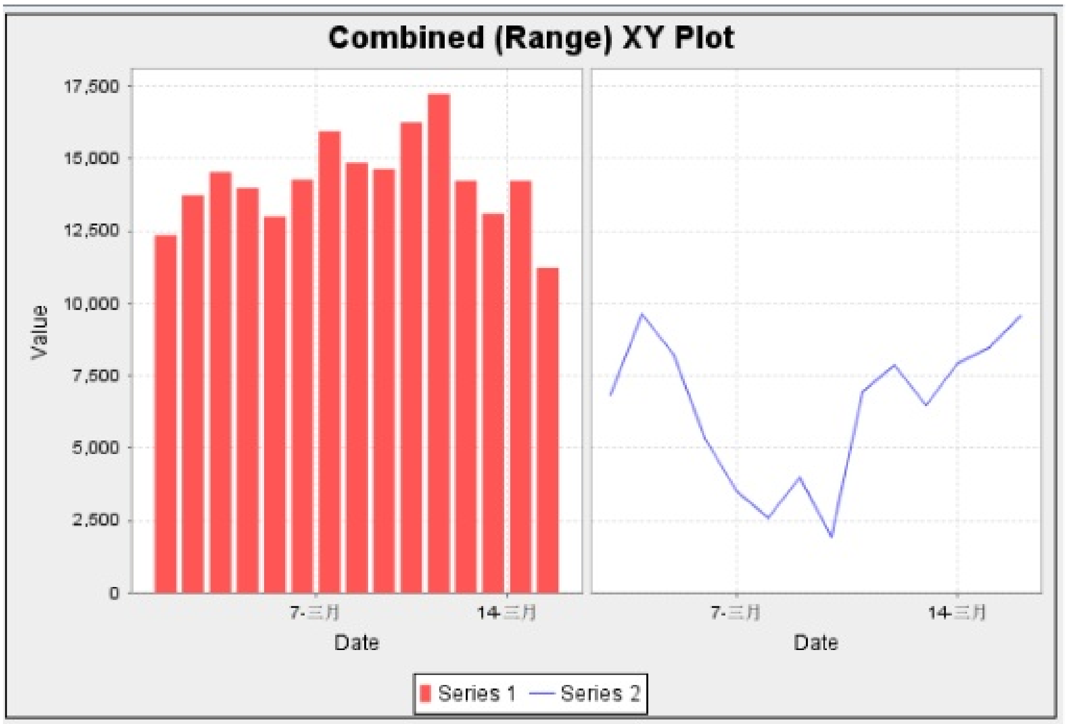


图2.26 共用相同范围轴的复合图表（参考：CombinedXYPlotDemo2.java）

2.14 开发远景

JFreeChart为免费软件，任何人可以扩展它，可以添加新的特征。已经有80多人向JFreeChart项目贡献代码。不久开发者将开发更多的图表来满足更多的需求。我们可以从JFreeChart网站上获得更多的信息和版本的更新：

http://www.jfree.org/jfreechart/

欢迎您的加入。

3 下载和安装JFreeChart 1.0.6

3.1 简介

本章主要介绍JFreeChart的下载、解包和编译内容。同时讲述如何运行JfreeChart应用实例，如何从源代码生成JavaDoc的HTML文件内容等。

3.2 下载

我们可以从JFreeChart主网站上获得最新的JFreeChart版本：

http://www.jfree.org/jfreechart/download

网页上提供了两个可下载的版本：

文件	描述
jfreechart-1.0.6.tar.gz	Linux/Unix版本
jfreechart-1.0.6.zip	Windows版本

这两个文件包含相同的源代码。主要不同在于下载的zip文件中所有的文本文件已被重新编码，有回车返回，并且每行结尾有换行符号。

JFreeChart是有JCommon（目前版本是1.0.9）基础类库.JCommon是运行时的jar文件，在JFreeChart下载时，包含该jar文件。如果你需要JCommon的源代码，可以从下面链接下载：

http://www.jfree.org/jcommon/

3.3 解包

下载完JFreeChart压缩文件之后，我们需要将该压缩文件解开。我们可以将文件解压到指定的目录下面。

3.3.1 Linux/Unix环境下解压

在Linux/Unix环境下解压文件，使用下面命令：

```
tar xvzf jfreechart-1.0.6.tar.gz tar xvzf jfreechart-1.0.6.tar.gz
```

该命令将JFreeChart所有的文件包括源代码，运行jar文件和doc文档解压到jfreechart-1.0.6目录下面。

3.3.2 Windows环境下解压

在Windows环境下解压文件，使用下面命令：

```
jar -xvf jfreechart-1.0.6.zip
```

该命令将JFreeChart所有的文件包括源代码，运行jar文件和doc文档解压到jfreechart-1.0.6目录下面。

3.3.3 文件目录说明

解压后的jfreechart-1.0.6目录下面，有许多文件和文件夹，列表如下：

文件/目录	说明
Ant	该目录下面包含了一个ant的build.xml脚本。我们使用该脚本可以用现有的版本源代码重新构建JFreeChart。
checkstyle	该目录下面包含了几种检查风格属性文件。文件定义了JFreeChart中的编码规范。
experimental	该文件夹下面包含了一些不属于JFreeChart标准API的类文件。注意这些代码的API可能会改变。
gjdoc	该文件夹下面包含了一种产生JFreeChart文档的脚本。
lib	该目录下面包含了JFreeChart的jar文件，以及JFreeChart依赖的jar文件。
source	JFreeChart源代码目录。
swt	该目录下面包含了具有实践经验的swt源代码。注意该代码API有可能发生改变。
Tests	JFreeChart单元测试的源代码文件。
jfreechart-1.0.6-demo.jar	一个具有实例演示的可运行的jar文件。
CHANGELOG.txt	老的JFreeChart变更的日志记录。
ChangeLog	JFreeChart变更的详细日志记录。
licence-LGPL.txt	JFreeChart公共认证（GNU LGPL）
NEWS	JFreeChart项目新闻
README.txt	重要信息—一定要读！

我们应当花一部分时间来熟悉这些文件，并详细的阅读README.txt文件。

3.4 运行演示实例

在解压的文件中，有一个实例演示应用包含了JFreeChart产生的大量图表演示实例。输入下面命令可以运行该应用：

```
java -jar jfreechart-1.0.6-demo.jar
```

实例的源代码跟JFreeChart开发指南一并发行，是收费的。

3.5 编译源代码

我们可是使用ant的build.xml文件重新编译JFreeChart类文件。进入ant目录，然后键入：

```
ant complie
```

这将重新编译全部的源文件和创建创建JFreeChart运行时依赖的jar文件。Ant工具需要1.5.1版本或更高版本，我们可以从下面链接获得更多ant信息：

http://ant.apache.org/

3.6 产生javadoc文档

JFreeChart源代码文件中包含了大量丰富的Javadoc注释。我们使用javadoc工具可以直接从源代码中产生HTML文件。

产生javadoc文件使用ant的javadoc目标，进入ant目录键入：

```
ant javadoc
```

这将产生javadoc目录。目录下面包含了全部的Javadoc的HTML文件。

4 使用JFreeChart1.0.6

4.1 概述

本章节向使用JFreeChart的新用户编写了一个简单的实例，以说明JFreeChart的使用方法。

4.2 创建第一个图表

4.2.1 概述

使用JFreeChart创建图表共有三个步骤。如下：

- 创建一个dataset。该dataset包含图表要显示的数据。
- 创建一个JFreeChart对象。该对象负责画这个图表。
- 创建一个输出目标（如：一个panel，显示在屏幕上）。该输出目标画这个图表。

下面，我们使用一个简单的应用（First.java）来描述这个过程。该应用产生了一个饼图，如下图4.1所示：

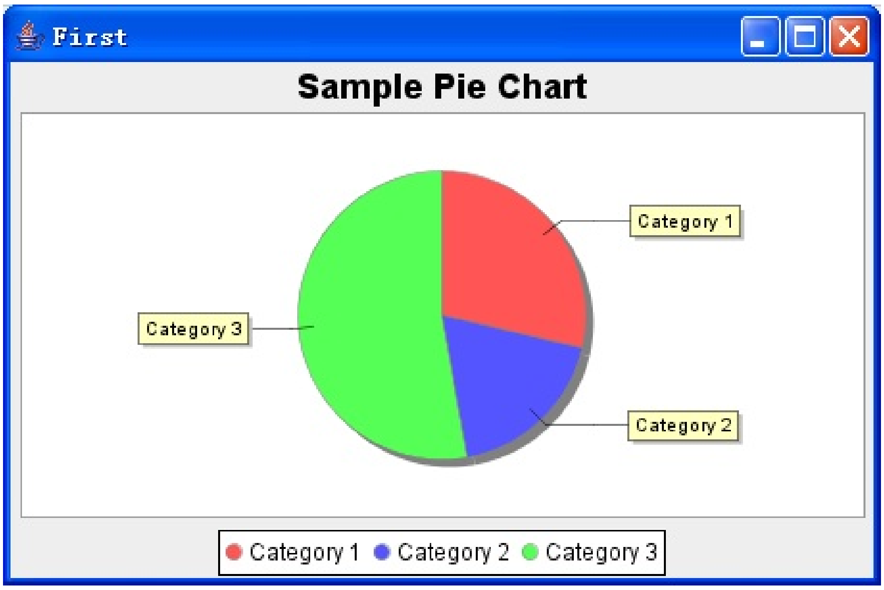


图4.1 创建的第一个饼图（参考First.java）

上面描述的三个步骤，将在下面的章节里面，均有代码详细说明。

4.2.2 数据

步骤一要求我们为我们的图表创建一个dataset。使用DefaultPieDataset类可以很容易创建。如下代码：

```
// create a dataset...
DefaultPieDataset dataset = new DefaultPieDataset();
dataset.setValue("Category 1", 43.2);
dataset.setValue("Category 2", 27.9);
dataset.setValue("Category 3", 79.5);
```

注意：

JFreeChart可以使用符合PieDataset接口的任何实现数据来创建饼图。DefaultDataset类实现了PieDataset接口，提供了一种便利的使用方式。

我们可以自由的开发符合实际需的任意PieDataset接口实现。

4.2.3 创建一个饼图

步骤二关心的是我们如何使用这个dataset展示在区域中。这就需要我们创建一个JFreeChart对象，该对象使用我们的饼图dataset数据画一个图表。我们使用ChartFactory类来创建，代码如下：

```
// create a chart...
JFreeChart chart = ChartFactory.createPieChart(
    "Sample Pie Chart",
    dataset,
    true, // legend?
    true, // tooltips?
    false // URLs?
);
```

注意：

代码中将一个dataset的引用传入到工厂方法中。JFreeChart持有这个dataset引用的目的是便于在画图表时能够获得数据。使用JFreeChart创建图表有许多定制外观的方式，在这个例子中我们使用缺省的属性值。后面章节将详细介绍。

4.2.4 显示图表

最好一个步骤就是在某个地方显示该图表。JFreeChart提供了非常灵活的图表输出方式。

现在我们可以在一个屏幕的框架中显示这个图表。ChartFrame具有显示图表的机制（ChartPanel）。代码如下：

```
// create and display a frame...
ChartFrame frame = new ChartFrame("First", chart);
frame.pack();
frame.setVisible(true);
```

代码全部完成，运行main()方法,可以出现图4.1界面。

4.2.5 全部程序代码

下面是整个例子的全部代码，更加清楚的看到我们需要导入的类包和实现方法。

```
public class First {
    public static void main(String[] args) {
        // create a dataset...
        DefaultPieDataset dataset = new DefaultPieDataset();
        dataset.setValue("Category 1", 43.2);
        dataset.setValue("Category 2", 27.9);
        dataset.setValue("Category 3", 79.5);
        // create a chart...
        JFreeChart chart = ChartFactory.createPieChart(
            "Sample Pie Chart",
            dataset,
            true, // legend?
            true, // tooltips?
            false // URLs?
        );
        // create and display a frame...
        ChartFrame frame = new ChartFrame("First", chart);
        frame.pack();
        frame.setVisible(true);
    }
}
```

5 饼图（Pie Charts）

5.1 简介

本章主要讲解JFreeChart中饼图的一些特征。内容如下：

- 控制颜色和饼图片区的外廓
- null值和零值的处理
- 饼图片区的标签（定制文本，改变分配的比例空间）
- “取出”某个片区
- 多个饼图显示
- 显示3D效果的饼图

更多的信息，可以参见PiePlot参考文档，见章节33.72。

5.2 创建一个简单的饼图（Pie Charts）

在前面的第四章的手把手的向导里创建了一个简单的饼图。在这里不做详细介绍。

5.3 片区颜色

饼图片区缺省填充的颜色是自动分配的，正如你上面实例看到的。如果你不喜欢这个缺省的颜色，你可以实用setSectionPaint（）方法来设置片区颜色。例如：

```
PiePlot plot = (PiePlot) chart.getPlot();
plot.setSectionPaint("Section A", new Color(200, 255, 255));
plot.setSectionPaint("Section B", new Color(200, 200, 255));
```

JFreeChart的实例PieChartDemo2.java演示了如何定制颜色。在JFreeChart的代码中，片区颜色使用三层色属性机制来定义的。同时，我们也可以对饼图中的每一个系列定义填充的颜色，这里我们不做细述，更多的信息请参阅PiePlot类（33.27章节）。

5.4 片区外廓

每一个饼图片区的外廓默认是一条细灰线勾画出来的。PiePlot类提供了如下选择项：

- 完全不显示片区外廓
- 通过改变缺省的值来改变全部的片区外廓
- 单独改变部分饼图的片区外廓

5.4.1 片区外廓的可见性控制

为了完全关闭片区外廓，使用下面代码：

```
PiePlot plot = (PiePlot) chart.getPlot();
plot.setSectionOutlinesVisible(false);
```

在任何时候，你只需要使用下面代码可以让外廓显示出来：

```
plot.setSectionOutlinesVisible(true);
```

调用该方法可以触发PlotChangeEvent事件。

5.4.2 片区外廓的控制

在片区外廓显示的时候，我们可以改变饼图片区的整个外廓颜色或风格或者单个饼图片区的颜色或风格。整个外廓颜色或风格的修改需要在基本层里面设置，单个饼图片区的颜色设置需要在系列层中设置。在基本层里，如果没有更高层的颜色设置，则调用已定义的默认设置。我们可以使用PiePot类的方法来改变我们的设置。如下方法：

```
public void setBaseSectionOutlinePaint(Paint paint);
public void setBaseSectionOutlineStroke(Stroke stroke);
```

有时候在图表里面，我们会更喜欢设置饼图里面某个具体的片区的外廓的颜色，或许突出显示某些片区的细节方面。做到这些，我们可以是使用系列层层设置，通过下面的方法来定义。

```
public void setSectionOutlinePaint(Comparable key, Paint paint);
public void setSectionOutlineStroke(Comparable key, Stroke stroke);
```

方法的第一个参数是dataset的片区关键值。如果我们将该值设为null，则系统将使用基本层的设置。

5.5 空置、零值和负值

PieDataset可能会包含一些饼图不可能显示的数值，比如null、零值或者负值。对于这些数据PiePlot类有专门的处理机制来处理。如果是零值，并且该值有意义，PiePlot类默认将一个标签放置在饼图片区显示的位置，并且在图表的图例中添加一个分类。如果零值可以忽略，我们可以使用下面代码设置一个标志，不显示该数据：

```
PiePlot plot = (PiePlot) chart.getPlot();
plot.setIgnoreZeroValues(true);
```

类似的null值也是如此处理，nul值代表dataset丢失或者不知来源的值。缺省的处理与零值相同，如果忽略null值，则代码如下：

```
PiePlot plot = (PiePlot) chart.getPlot();
plot.setIgnoreNullValues(true);
```

在饼图中处理负值是非常不明知的，所以在JFreeChart中负值总是被忽略的。

5.6 片区和图例标签

片区标签使用的文本，即可以在图表上显示，也可以在图表的图例上显示，并且完全可以定制。标签是自动默认产生的，但我们可以使用下面方法来改变：

```
public void setLabelGenerator(PieSectionLabelGenerator generator);
public void setLegendLabelGenerator(PieSectionLabelGenerator genera
```

StandPieSectionLabelGenerator类专门用来生成图例的一个实现类，提供灵活处理定制标签的功能（如果你不喜欢用这个类，可以定义自己的类，只要实现接口PieSectionLabelGenerator即可）。Dataset显示出的标签值由Javade信息格式类来进行格式化——表5.1所示格式化的变量值。

名称	描述
{0}	片区关键值（字符串）
{1}	片区值
{2}	百分比的片区值

表5.1 StandardPieSectionLabelGenerator substitutions

下面举例说，假如我们有一个PieData包含下面的值

片区标识	片区值
S1	3.0
S2	5.0
S3	Null
S4	2.0

表 5.2 一个dataset实例

下面是格式化字符串产生的标签值内容：

格式化字符串	片区	产生的标签值
{0}	0	S1
{0} has value {1}	1	S2 has value 5.0
{0}({2} percent)	0	S1(30 percent)
{0} = {1}	2	S3 = null

类PieChartDemo2.java使用了定制标签的方法。

5.7 “取出”某个片区

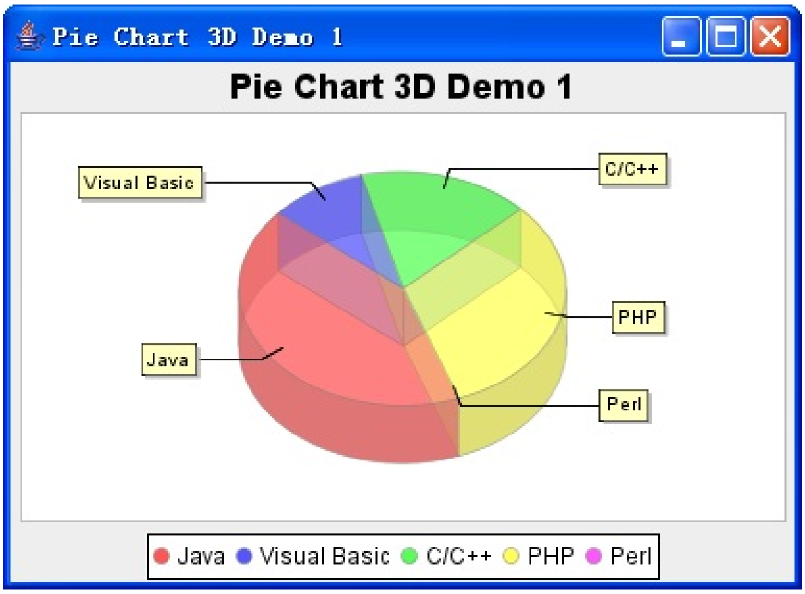
PiePlot类支持将某个片区“取出“显示。即某个片区偏离图表中心，以突出显示。如类所显示。

片区偏离的数值是图表半径的一个百分值来表示。例如0.3（30 persent）代码偏离的值是半径的长度×0.3.代码如下：

```
PiePlot pieplot = (PiePlot) jfreechart.getPlot();
pieplot.setExplodePercent("Two", 0.5);
```

5.8 3D饼图

JFreeChart具有一个实现3D效果的饼图类PiePlot3D，如图5.5所示，(PieChart3DDemo1.java)。PiePlot3D是PiePlot的子类，因此在我们创建自己的饼图时，使用PiePlot3D替换掉原来的PiePlot即可。创建3D效果的饼图时，使用ChartFactory的 createPieChart3D（）方法，而不是createPieChart（）方法。



如5.5 3D效果图。

对于该类有一些限制，如下：

- 不支持"取出"片区功能。
- 不支持轴项转动——如果支持，3D效果图可能会变型。

3D的实例主要是类PieChart3DDemo1-3.java。讲解类中没有列出其他两个。因为功能雷同于非3D效果。

5.9 多饼图

我们可是使用类MultiplePiePlot在一个图表上显示多个饼图。饼图的数据使用CatoryDataset。如图5.6所示。每个独立的饼图由一个专门的图表多次创建而成。创建的每一个饼图的PieDataset是由系统提供的CategoryDataset按照行或者列拆分出来的。代码见5.10.11.

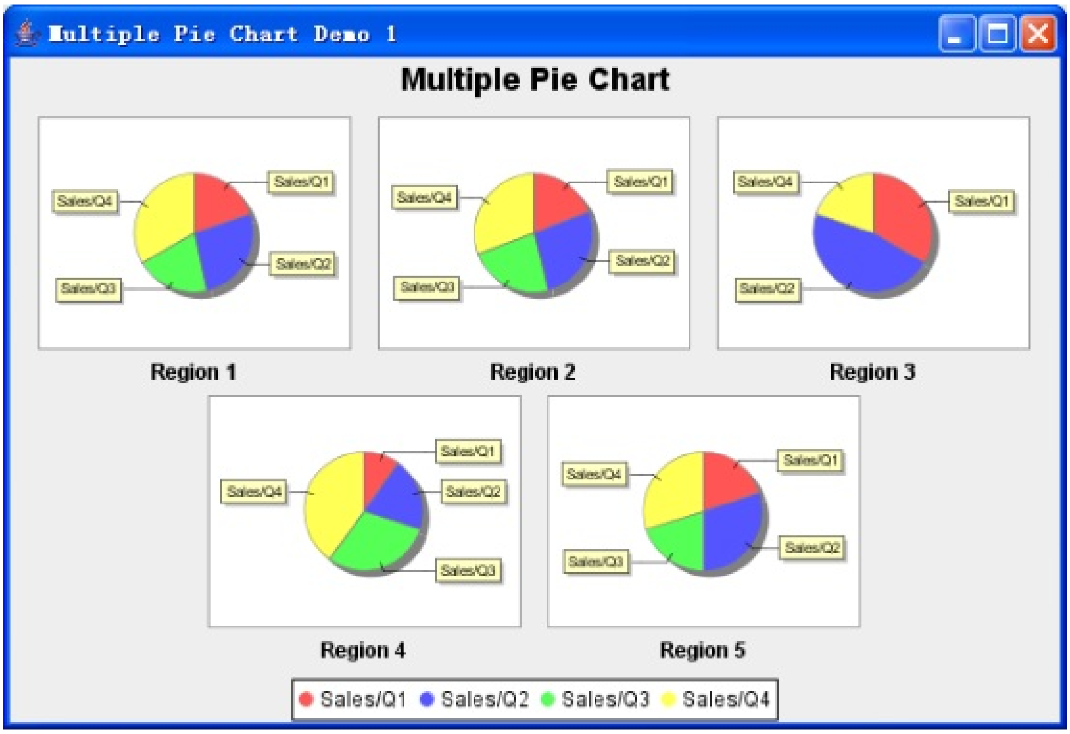


图5.6 多饼图图表

5.10 实例讲解

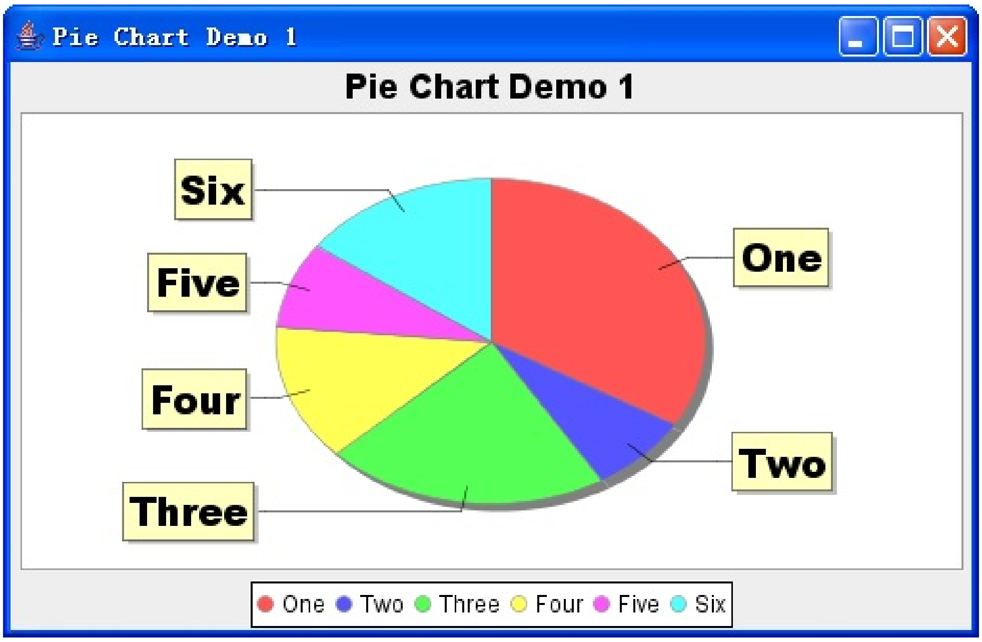
5.10.1 体会

也分为三层界面显示层、数据层、控制层。

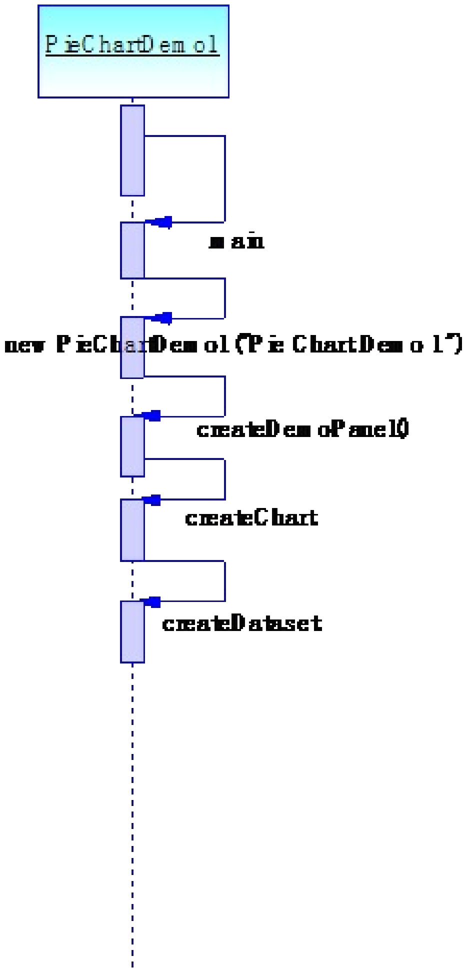
- 数据层比较复杂。每个图形有不同的数据类型。Dataset类控制。
- 控制层Plot，饼图使用PiePlot设置显示的图形的面貌的控制，JFreeChart最丰富的功能。
- 设置片区颜色
- 设置标签（文本格式、背景颜色等）
- 设置是否取出某块。
- 设置饼图是否为圆形
- 设置饼图旋转。
- 显示层JFreeChart使用数据、控制直接将数据显示出来。

5.10.2 类PieChartDemo1.java

效果图如下：



代码编写典型的使用了面对对象的方法：



全部代码如下：

```
package demo;
import java.awt.Dimension;
import java.awt.Font;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PiePlot;
import org.jfree.chart.title.TextTitle;
import org.jfree.data.general.DefaultPieDataset;
import org.jfree.data.general.PieDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;
public class PieChartDemo1 extends ApplicationFrame {
    /**
    *
    */
    private static final long serialVersionUID = 25985575577240854
    public PieChartDemo1(String string) {
        super(string);
        JPanel jpanel = createDemoPanel();
        jpanel.setPreferredSize(new Dimension(500, 270));
        setContentPane(jpanel);
```

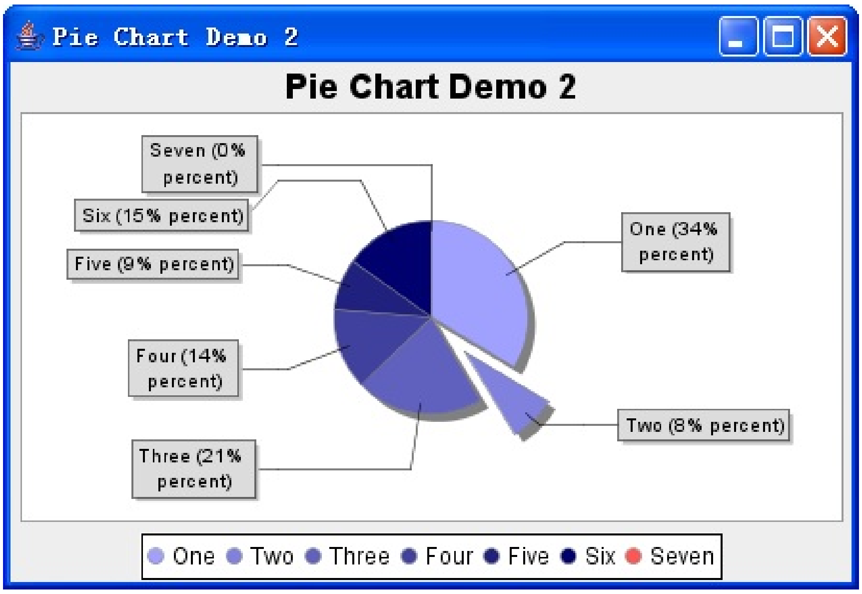
```
    }
    private static PieDataset createDataset() {
        DefaultPieDataset defaultpiedataset = new DefaultPieDataset
        defaultpiedataset.setValue("One", new Double(43.2));
        defaultpiedataset.setValue("Two", new Double(10.0));
        defaultpiedataset.setValue("Three", new Double(27.5));
        defaultpiedataset.setValue("Four", new Double(17.5));
        defaultpiedataset.setValue("Five", new Double(11.0));
        defaultpiedataset.setValue("Six", new Double(19.4));
        return defaultpiedataset;
    }
    private static JFreeChart createChart(PieDataset piedataset) {
        JFreeChart jfreechart = ChartFactory.createPieChart("Pie Ch
        piedataset, true, true, false);
        TextTitle texttitle = jfreechart.getTitle();
        texttitle.setToolTipText("A title tooltip!");
        PiePlot pieplot = (PiePlot) jfreechart.getPlot();
        pieplot.setLabelFont(new Font("Arial Black", 0, 20));
        pieplot.setNoDataMessage("No data available");
        pieplot.setCircular(false);
        pieplot.setLabelGap(0.02);
        return jfreechart;
    }
    public static JPanel createDemoPanel() {
        JFreeChart jfreechart = createChart(createDataset());
        return new ChartPanel(jfreechart);
    }
    public static void main(String[] strings) {
        PieChartDemo1 piechartdemo1 = new PieChartDemo1("Pie Chart
        piechartdemo1.pack();
        RefineryUtilities.centerFrameOnScreen(piechartdemo1);
        piechartdemo1.setVisible(true);
    }
}
```

5.10.3 类PieChartDemo2.java

功能：

“取出“片区显示。

效果：



代码：

```
private static JFreeChart createChart(PieDataset piedataset) {
    JFreeChart jfreechart = ChartFactory.createPieChart("PieChart D
    PiePlot pieplot = (PiePlot) jfreechart.getPlot();
    pieplot.setSectionPaint("One", new Color(160, 160, 255));
    pieplot.setSectionPaint("Two", new Color(128, 128, 223));
    pieplot.setSectionPaint("Three", new Color(96, 96, 191));
    pieplot.setSectionPaint("Four", new Color(64, 64, 159));
    pieplot.setSectionPaint("Five", new Color(32, 32, 127));
    pieplot.setSectionPaint("Six", new Color(0, 0, 111));
    pieplot.setNoDataMessage("No data available");
    pieplot.setExplodePercent("Two", 0.5);
    pieplot.setLabelGenerator(new StandardPieSectionLabelGenerator(
    pieplot.setLabelBackgroundPaint(new Color(220, 220, 220));
    pieplot.setLegendLabelToolTipGenerator(new StandardPieSectionLa
    return jfreechart;
}
```

程序代码说明：

- `setSectionPaint("One", new Color(160, 160, 255))`：设置某个片区的填充颜色。第一个参数为片区的标识，第二个参数为色值。
- `setNoDataMessage("No data available")`：设置dataset为null时显示的提示信息。
- `setLabelGenerator(new StandardPieSectionLabelGenerator("{0}({2}`：设置标签显示的格式。
- `setLabelBackgroundPaint(new Color(220, 220, 220))`：设置标签的背景颜色。
- `setLegendLabelToolTipGenerator(new StandardPieSectionLabelGenera`：设置鼠标滑过图表是显示鼠标当前片区的提示信息。
- `pieplot.setExplodePercent("Two", 0.5)`：将第2个片区取出显示。后面

一个参数是取出的距离，是一个比例数。

5.10.4 类 PieChartDemo3.java

功能：

显示了dataset为空时，设置的提示信息。

效果：



代码：

```
private static JFreeChart createChart(PieDataset piedataset) {
    JFreeChart jfreechart = ChartFactory.createPieChart("Pie Chart
    null, true, true, false);
    PiePlot pieplot = (PiePlot) jfreechart.getPlot();
    pieplot.setNoDataMessage("没有有效的数据显示!");
    pieplot.setNoDataMessageFont(new Font("黑体", 2, 20));
    pieplot.setNoDataMessagePaint(Color.red);
    return jfreechart;
}
```

程序代码说明：

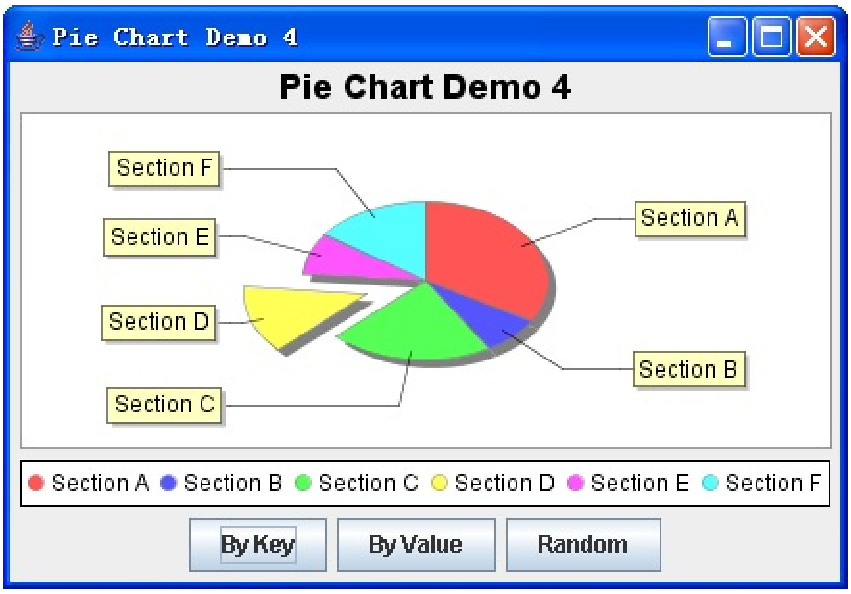
- `setNoDataMessage("没有有效的数据显示!")` ：设置提示信息内容。
- `setNoDataMessageFont(new Font("Serif", 2, 10))` ：设置提示信息的字体和大小。
- `setNoDataMessagePaint(Color.red)` ：设置提示信息字体的颜色。

5.10.5 类 PieChartDemo4.java

功能：

带有按钮的图表，通过对dataset的排序，可以改变片区的位置。

效果：



代码：

```
public void actionPerformed(ActionEvent actionevent) {
    String string = actionevent.getActionCommand();
    if ("BY_KEY".equals(string)) {
        if (!ascendingByKey) {
            dataset.sortByKeys(SortOrder.ASCENDING);
            ascendingByKey = true;
        } else {
            dataset.sortByKeys(SortOrder.DESCENDING);
            ascendingByKey = false;
        }
    } else if ("BY_VALUE".equals(string)) {
        if (!ascendingByValue) {
            dataset.sortByValues(SortOrder.ASCENDING);
            ascendingByValue = true;
        } else {
            dataset.sortByValues(SortOrder.DESCENDING);
            ascendingByValue = false;
        }
    } else if ("RANDOM".equals(string)) {
        ArrayList arraylist = new ArrayList(dataset.getKeys());
        Collections.shuffle(arraylist);
        DefaultPieDataset defaultpiedataset = new DefaultPieDataset
        Iterator iterator = arraylist.iterator();
        while (iterator.hasNext()) {
            Comparable comparable = (Comparable) iterator.next();
            defaultpiedataset.setValue(comparable, dataset
            .getValue(comparable));
        }
        PiePlot pieplot = (PiePlot) chart.getPlot();
        pieplot.setDataset(defaultpiedataset);
        dataset = defaultpiedataset;
    }
}
```

程序代码说明：

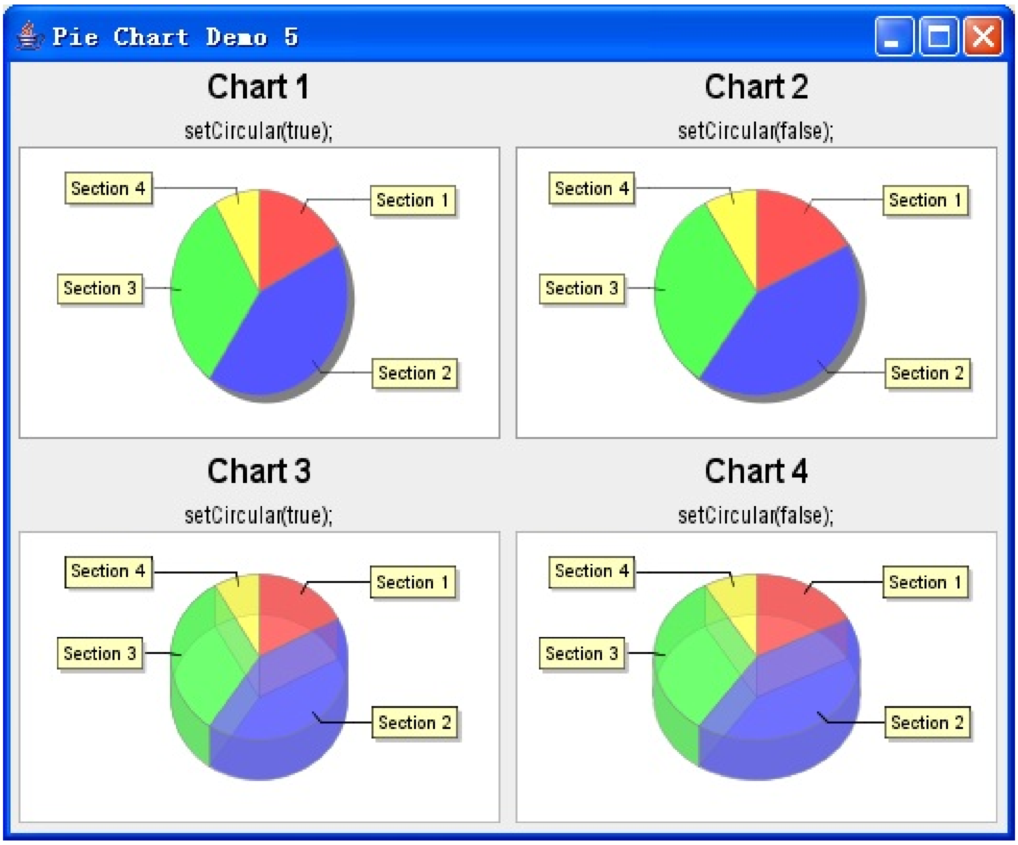
- `dataset.sortByKeys(SortOrder.ASCENDING);` 通过片区的关键值进行升序排序。
- `dataset.sortByValues(SortOrder.DESCENDING);` 通过片区的值进行降序排序。

5.10.6 类PieChartDemo5.java

功能：

右边两个饼图为原形的，而左边两个为椭圆形的。

效果：


Pie Chart Demo 5
Chart 1
setCircular(true);
Section 4
Section 1
Section 3
Section 2
Chart 2
setCircular(false);
Section 4
Section 1
Section 3
Section 2
Chart 3
setCircular(true);
Section 4
Section 1
Section 3
Section 2
Chart 4
setCircular(false);
Section 4
Section 1
Section 3
Section 2

代码：

```
public static JPanel createDemoPanel() {
    JPanel jpanel = new JPanel(new GridLayout(2, 2));
    DefaultPieDataset defaultpiedataset = new DefaultPieDataset();
    defaultpiedataset.setValue("Section 1", 23.3);
    defaultpiedataset.setValue("Section 2", 56.5);
    defaultpiedataset.setValue("Section 3", 43.3);
    defaultpiedataset.setValue("Section 4", 11.1);
    //
    JFreeChart jfreechart = ChartFactory.createPieChart("Chart 1",
        defaultpiedataset, false, false, false);
    jfreechart.addSubtitle(new TextTitle("setCircular(true);", new
        "Dialog", 0, 12)));
    PiePlot pieplot = (PiePlot) jfreechart.getPlot();
    pieplot.setCircular(true);
    //
    JFreeChart jfreechart_0_ = ChartFactory.createPieChart("Chart 2
        defaultpiedataset, false, false, false);
    jfreechart_0_.addSubtitle(new TextTitle("setCircular(false);",
        new Font("Dialog", 0, 12)));
    PiePlot pieplot_1_ = (PiePlot) jfreechart_0_.getPlot();
    pieplot_1_.setCircular(false);
    //
    JFreeChart jfreechart_2_ = ChartFactory.createPieChart3D("Chart
        defaultpiedataset, false, false, false);
    jfreechart_2_.addSubtitle(new TextTitle("setCircular(true);", n
        "Dialog", 0, 12)));
    PiePlot3D pieplot3d = (PiePlot3D) jfreechart_2_.getPlot();
    pieplot3d.setForegroundAlpha(0.6F);
    pieplot3d.setCircular(true);
    //
    JFreeChart jfreechart_3_ = ChartFactory.createPieChart3D("Chart
        defaultpiedataset, false, false, false);
    jfreechart_3_.addSubtitle(new TextTitle("setCircular(false);",
        new Font("Dialog", 0, 12)));
    PiePlot3D pieplot3d_4_ = (PiePlot3D) jfreechart_3_.getPlot();
    pieplot3d_4_.setForegroundAlpha(0.6F);
    pieplot3d_4_.setCircular(false);
    jpanel.add(new ChartPanel(jfreechart));
    jpanel.add(new ChartPanel(jfreechart_0_));
    jpanel.add(new ChartPanel(jfreechart_2_));
    jpanel.add(new ChartPanel(jfreechart_3_));
    jpanel.setPreferredSize(new Dimension(800, 600));
    return jpanel;
}
```

程序代码说明：

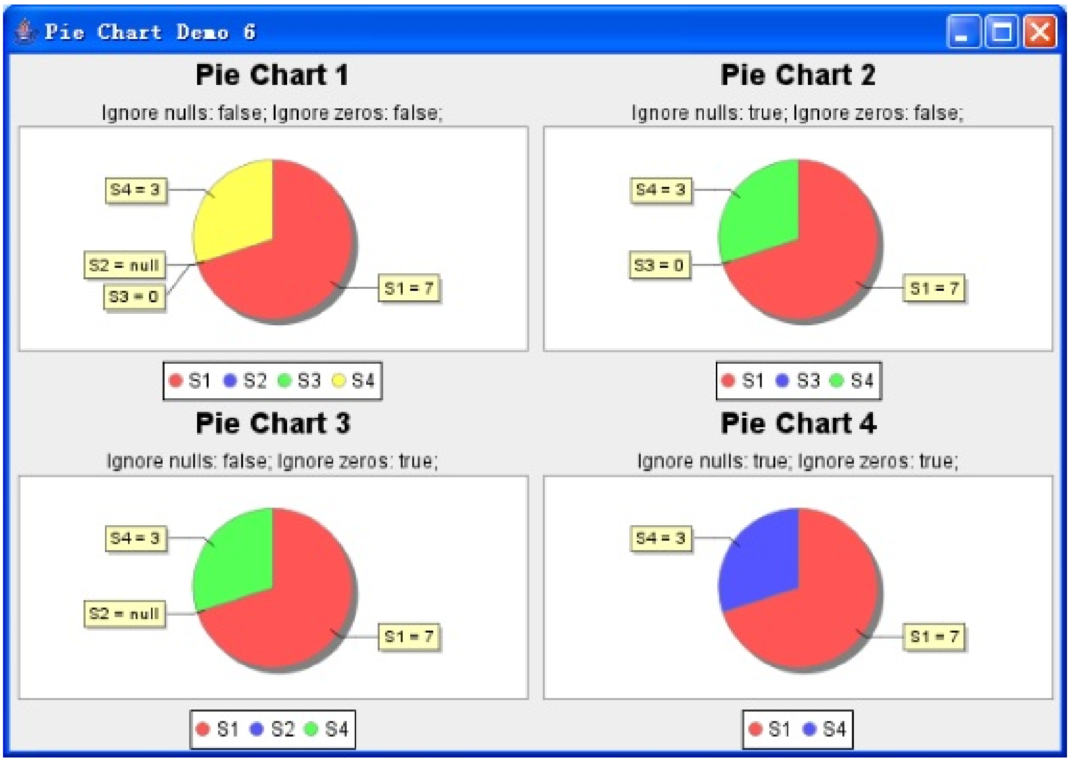
- `pieplot.setCircular(true)` ：设置饼图为圆形。
- `jpanel.add` 方法可以添加多个图形的panel

5.10.7 类PieChartDemo6.java

功能：

显示饼图对零值和null值的处理。

效果：



代码：

```
public static JPanel createDemoPanel() {
JPanel jpanel = new JPanel(new GridLayout(2, 2));
    JFreeChart jfreechart = createChart("Pie Chart 1", createDatase
    Font font = new Font("Dialog", 0, 12);
    jfreechart.addSubtitle(new TextTitle(
        "Ignore nulls: false; Ignore zeros: false;", font));
    JFreeChart jfreechart_0_ = createChart("Pie Chart 2", createDat
    jfreechart_0_.addSubtitle(new TextTitle(
        "Ignore nulls: true; Ignore zeros: false;", font));
    PiePlot pieplot = (PiePlot) jfreechart_0_.getPlot();
    pieplot.setIgnoreNullValues(true);
    pieplot.setIgnoreZeroValues(false);
    JFreeChart jfreechart_1_ = createChart("Pie Chart 3", createDat
    jfreechart_1_.addSubtitle(new TextTitle(
        "Ignore nulls: false; Ignore zeros: true;", font));
    PiePlot pieplot_2_ = (PiePlot) jfreechart_1_.getPlot();
    pieplot_2_.setIgnoreNullValues(false);
    pieplot_2_.setIgnoreZeroValues(true);
    JFreeChart jfreechart_3_ = createChart("Pie Chart 4", createDat
    jfreechart_3_.addSubtitle(new TextTitle(
        "Ignore nulls: true; Ignore zeros: true;", font));
    PiePlot pieplot_4_ = (PiePlot) jfreechart_3_.getPlot();
    pieplot_4_.setIgnoreNullValues(true);
    pieplot_4_.setIgnoreZeroValues(true);
    jpanel.add(new ChartPanel(jfreechart));
    jpanel.add(new ChartPanel(jfreechart_0_));
    jpanel.add(new ChartPanel(jfreechart_1_));
    jpanel.add(new ChartPanel(jfreechart_3_));
    return jpanel;
}
```

程序代码说明：

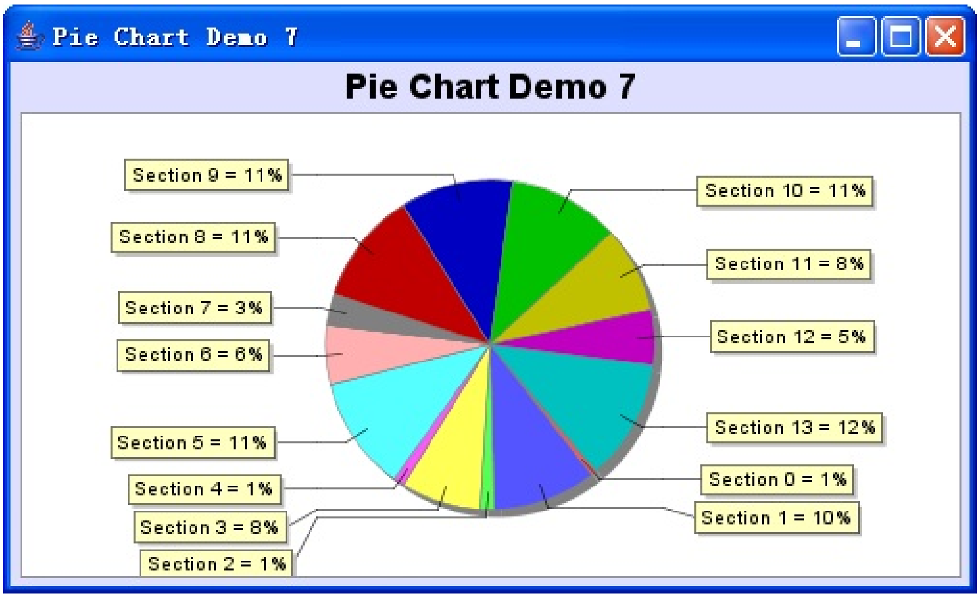
- `pieplot.setIgnoreNullValues(true)` ：设置饼图忽略null值，即是null值将不显示。
- `pieplot.setIgnoreZeroValues(false);` 设置饼图不忽略零值。即图表中显示出零值。

5.10.8 类PieChartDemo7.java

功能：

图表能够旋转，标签也随之移动。

效果：



代码：

```
public static JPanel createDemoPanel() {
    PieDataset piedataset = createDataset(14);
    JFreeChart jfreechart = ChartFactory.createPieChart("Pie Chart
        piedataset, false, true, false);
    jfreechart.setBackgroundPaint(new Color(222, 222, 255));
    PiePlot pieplot = (PiePlot) jfreechart.getPlot();
    pieplot.setBackgroundPaint(Color.white);
    pieplot.setCircular(true);
    pieplot.setLabelGenerator(new StandardPieSectionLabelGenerator(
        "{0} = {2}", NumberFormat.getNumberInstance(), NumberFormat
        .getPercentInstance()));
    pieplot.setNoDataMessage("No data available");
    ChartPanel chartpanel = new ChartPanel(jfreechart);
    chartpanel.setPreferredSize(new Dimension(500, 270));
    Rotator rotator = new Rotator(pieplot);
    rotator.start();
    return chartpanel;
}
```

程序代码说明：

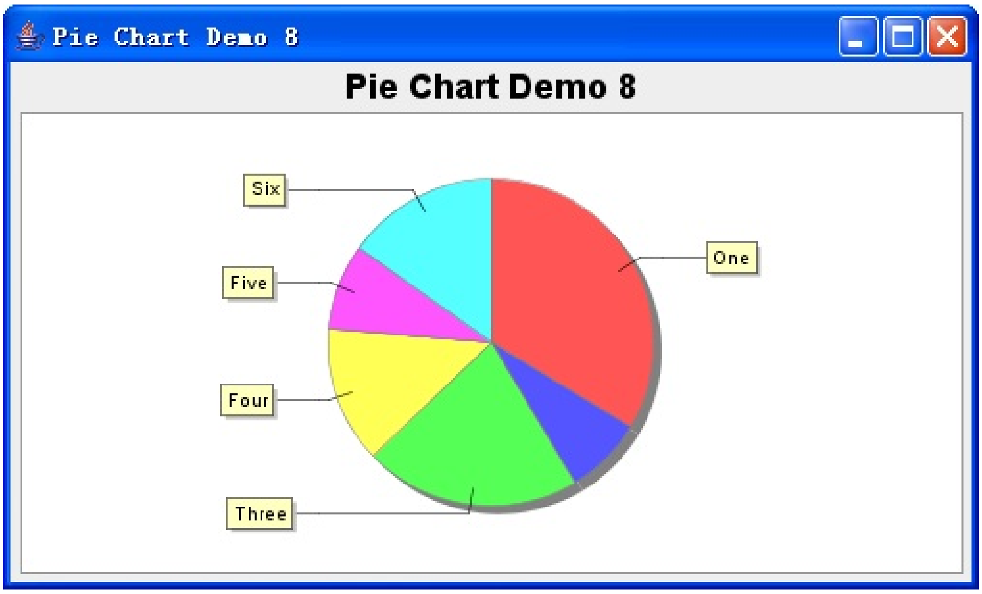
- 使用Rotator对象，旋转饼图。代码如上。

5.10.9 类PieChartDemo8.java

功能：

编写自定义标签产生器，将Two标签不显示。

效果：



代码：

```
private static JFreeChart createChart(PieDataset piedataset) {
    JFreeChart jfreechart = ChartFactory.createPieChart("Pie Chart
    piedataset, false, true, false);
    PiePlot pieplot = (PiePlot) jfreechart.getPlot();
    pieplot.setLabelGenerator(new CustomLabelGenerator());
    return jfreechart;
}
static class CustomLabelGenerator implements PieSectionLabelGenerat
    public String generateSectionLabel(PieDataset piedataset,
    Comparable comparable) {
    String string = null;
        if (piedataset != null && !comparable.equals("Two"))
            string = comparable.toString();
        return string;
    }
    public AttributedString generateAttributedSectionLabel(
        PieDataset piedataset, Comparable comparable) {
        Object object = null;
        String string = comparable.toString();
        String string_0_ = (string + " : " + String.valueOf(piedata
            .getValue(comparable)));
        AttributedString attributedstring = new AttributedString(st
        attributedstring.addAttribute(TextAttribute.WEIGHT,
        TextAttribute.WEIGHT_BOLD, 0, string.length() - 1);
        return attributedstring;
    }
}
```

程序代码说明：

- 自定义 `CustomLabelGenerator`，必须实现接

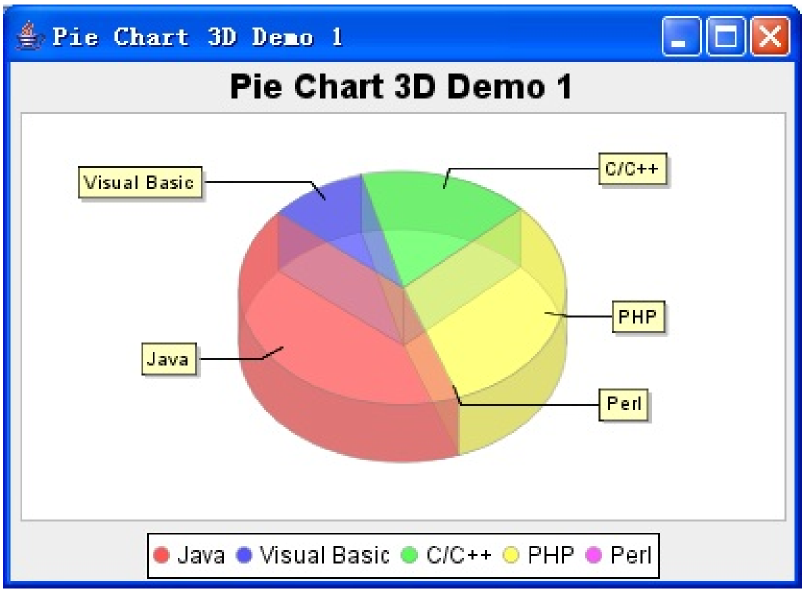
口 `PieSectionLabelGenerator` 。

5.10.10 类PieChart3DDemo1.java

功能：

3D效果的饼图。

效果：



代码：

```
private static JFreeChart createChart(PieDataset piedataset) {
    JFreeChart jfreechart = ChartFactory.createPieChart3D(
        "Pie Chart 3D Demo 1", piedataset, true, true, false);
    PiePlot3D pieplot3d = (PiePlot3D) jfreechart.getPlot();
    pieplot3d.setStartAngle(180.0);
    pieplot3d.setDirection(Rotation.CLOCKWISE);
    pieplot3d.setForegroundAlpha(0.5F);
    pieplot3d.setNoDataMessage("No data to display");
    return jfreechart;
}
```

程序代码说明：

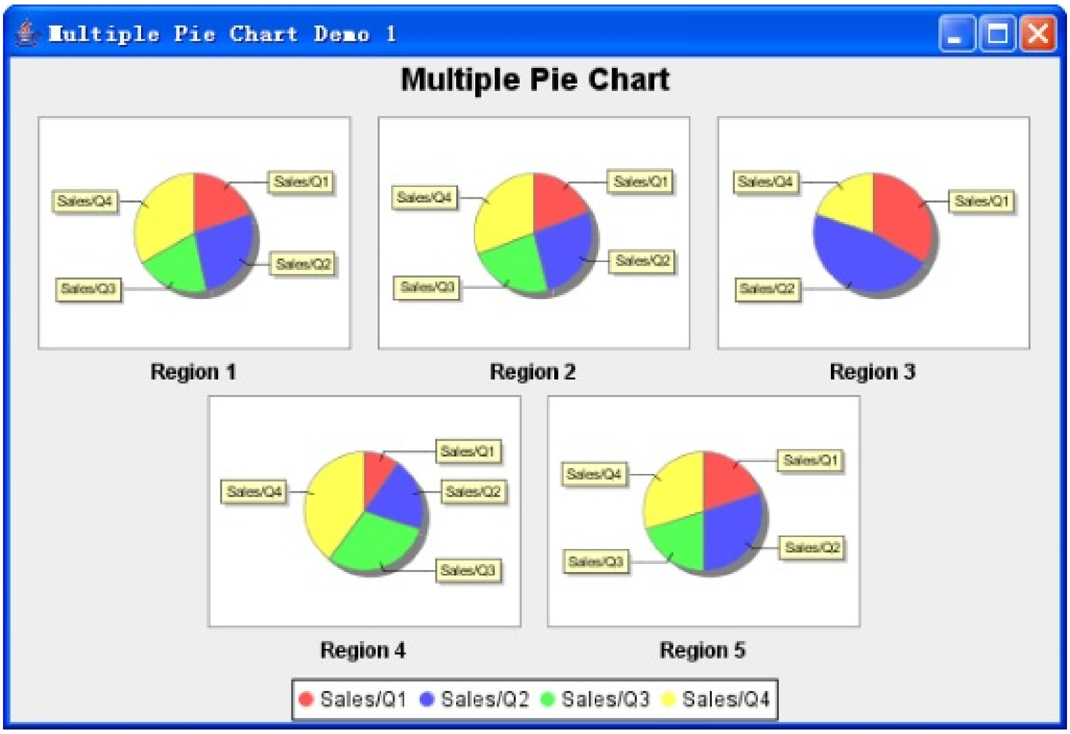
- 使用 `ChartFactory` 的方法 `createPieChart3D` 创建3D效果的饼图。
- `setStartAngle(180.0)` ：设置旋转角度。
- `setDirection(Rotation.CLOCKWISE)` ：设置旋转方向， `Rotation.CLOCKWISE` 为顺时针。
- `setForegroundAlpha(0.5F)` ：设置图表透明图0.0~1.0范围。0.0为完全透明，1.0为完全不透明。

5.10.11 类 MultiplePieChartDemo1.java

功能：

使用CategoryDataset数据集，在一个图表上产生多个饼图。

效果：



代码：

```
private static CategoryDataset createDataset() {
    double[][] ds = { { 3.0, 4.0, 3.0, 5.0 }, { 5.0, 7.0, 6.0, 8.0
        { 5.0, 7.0, Double.NaN, 3.0 }, { 1.0, 2.0, 3.0, 4.0 },
        { 2.0, 3.0, 2.0, 3.0 } };
    CategoryDataset categorydataset = DatasetUtilities
        .createCategoryDataset("Region ", "Sales/Q", ds);
    return categorydataset;
}
```

程序代码说明：

- 创建 `CategoryDataset` 的方法。

```
private static JFreeChart createChart(CategoryDataset categorydatas
    JFreeChart jfreechart = ChartFactory.createMultiplePieChart(
        "Multiple Pie Chart", categorydataset, TableOrder.BY_ROW, t
    MultiplePiePlot multiplepieplot = (MultiplePiePlot) jfreechart
    JFreeChart jfreechart_0_ = multiplepieplot.getPieChart();
    PiePlot pieplot = (PiePlot) jfreechart_0_.getPlot();
    pieplot.setLabelGenerator(new StandardPieSectionLabelGenerator(
    pieplot.setLabelFont(new Font("SansSerif", 0, 8));
    pieplot.setInteriorGap(0.3);
    return jfreechart;
}
```

程序代码说明：

- 使用 `ChartFactory` 的方法 `createMultiplePieChart()` 创建多个饼图的图表。
- `multiplepieplot.getPieChart()` ：获得单个饼图的图表。

6 直方条形图（Bar Charts）

6.1 简介

本章详细介绍了使用JFreeChart创建直方条形图的过程。开始我们先用一个简单的直方条形图例子进行说明，然后进一步深入了解JFreeChart为直方条形图提供的定制方法。在了解了这些标准的直方条形图配置项之后，然后再深入了解更复杂的图表：

- 堆栈式直方条形图
- 时序数据的条形直方图
- 柱状图

本章结束之后，我们将会对JFreeChart支持直方条形图创建的特点有个整体的了解。

6.2 创建一个直方条形图

6.2.1 概述

直方条形图常常被用来显示表列数据。如下表，为一个简单的两行、三列数据。

表 6.1

	Colnums1	Colnums2	Colnums3
Row1	1.0	5.0	3.0
Row2	2.0	3.0	2.0

在JFreeChart里，这个表格数据封装为一个dataset数据对象，每列标题为一个种类，每行为一个系列。每行标题为一个系列名称（或者系列关键值）。直方条形图展现的数据图如图6.2.

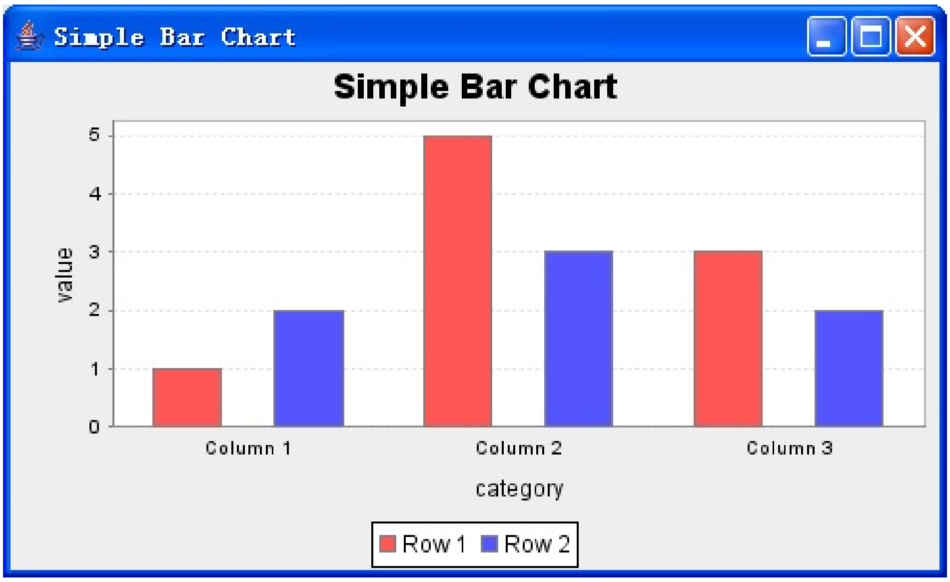


图6.2 简单的直方条形图（参见：BarExample1.java）

在这个图表的实例中，我们可以看到JFreeChart将每列数据（即一个种类）组合在一起。而且对每行数据（即每个系列）使用各种颜色高亮显示。图表的图例将颜色和系列的名称/关键值对应起来。

6.2.2 创一个dataset

创建直方条形图的第一步就是创建一个合适的dataset数据集。JFreeChart为直方条形图提供的访问表列数据的一系列方法，必须符合接口CategoryDataset定义。

JFreeChart中提供了一个便利的实现CategoryDataset接口的类为DefaultCategoryDataset。下面显示我们如何使用这个类来封装表6.1数据。代码如下：

```
private CategoryDataset createDataset() {
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    dataset.addValue(1.0, "Row 1", "Column 1");
    dataset.addValue(5.0, "Row 1", "Column 2");
    dataset.addValue(3.0, "Row 1", "Column 3");
    dataset.addValue(2.0, "Row 2", "Column 1");
    dataset.addValue(3.0, "Row 2", "Column 2");
    dataset.addValue(2.0, "Row 2", "Column 3");
    return dataset;
}
```

6.2.3 创建一个chart图表

接下来就是要创建一个JFreeChart的实例，使用上面提供的dataset数据集画一个直方条形图。简单的，我们使用ChartFactory类来创建这个JFreeChart实例。代码如下：

```
private JFreeChart createChart(CategoryDataset dataset) {
    JFreeChart chart = ChartFactory.createBarChart(
        "BarChartDemo", // chart title
        "Category", // domain axis label
        "Value", // range axis label
        dataset, // data
        PlotOrientation.VERTICAL, // orientation
        true, // include legend
        true, // tooltips?
        false // URLs?
    );
    return chart;
}
```

CreateBarChart()的大部分参数是比较容易理解的，但其中一部分还需要进一步说明。

- 图显示的方向可以是水平的，还是可以是垂直的。
- 图表的信息提示，是否要添加，有一个标志来控制——在上面的例子中，我们将这个标识设置为true，因此当我们在一个swing应用窗口显示这个图表时，我们会看到这个信息提示。
- URLs标志，设置为false。

我们完成这个直方条形图后，我们将会过头来，仔细看看ChartFactory类在后台做了写什么。

6.2.4 显示该chart图表

为了完成我们的第一个直方条形图实例，我们将JFreeChart实例传给一个ChartPanel对象，然后在一个Swing应用窗口上显示该实例。全部的代码如下：

```
import java.awt.Dimension;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;
/**
* A simple demonstration application showing how to create a bar ch
*/
public class BarExample1 extends ApplicationFrame {
    /**
    * Creates a new demo instance.
    *
    * @param title
    * the frame title.
    */
    public BarExample1(String title) {
        super(title);
        DefaultCategoryDataset dataset = new DefaultCategoryDataset
        dataset.addValue(1.0, "Row 1", "Column 1");
        dataset.addValue(5.0, "Row 1", "Column 2");
        dataset.addValue(3.0, "Row 1", "Column 3");
        dataset.addValue(2.0, "Row 2", "Column 1");
        dataset.addValue(3.0, "Row 2", "Column 2");
        dataset.addValue(2.0, "Row 2", "Column 3");
        JFreeChart chart = ChartFactory.createBarChart(
            "Bar Chart Demo", // chart
            // title
            "Category", // domain axis label
            "Value", // range axis label
            dataset, // data
            PlotOrientation.VERTICAL, // orientation
            true, // include legend
            true, // tooltips?
            false // URLs?
        );
        ChartPanel chartPanel = new ChartPanel(chart, false);
        chartPanel.setPreferredSize(new Dimension(500, 270));
        setContentPane(chartPanel);
    }
    /**
    * Starting point for the demonstration application.
    *
    * @param args
    * ignored.
    */
    public static void main(String[] args) {
```

```
        BarExample1 demo = new BarExample1("Bar Demo 1");
        demo.pack();
        RefineryUtilities.centerFrameOnScreen(demo);
        demo.setVisible(true);
    }
}
```

完成这些代码后，运行代码，将会显示如6.2图的界面。

6.3 ChartFactory类

在上面的实例代码中，我们使用ChartFactory类来组装一个JFreeChart实例来显示一个直方条形图。下面我们更仔细的看一下该类是如何工作的，因此我们可以看到直方条形图更多底层的框架结构。理解底层结构的关键是能定制图表的外观。下面是ChartFactory方法createBarChart()方法部分代码：

```
CategoryAxis categoryAxis = new CategoryAxis(categoryAxisLabel);
ValueAxis valueAxis = new NumberAxis(valueAxisLabel);
BarRenderer renderer = new BarRenderer();
……
CategoryPlot plot = new CategoryPlot(dataset, categoryAxis, valueAx
renderer);
plot.setOrientation(orientation);
JFreeChart chart = new JFreeChart(title, JFreeChart.DEFAULT_TITLE_F
```

以下就是代码所做的工作。

- 我们的直方条形图有两个轴，一个轴显示dataset（CategoryAxis）的种类，另一个是显示带有数据（NumberAxis）刻度的数据轴。上面代码中代码1、2行建立了这两个轴，轴的标签是createBarChart()方法传入的。
- 第三行，创建了一个BarRender——该类为每一个数据项目画直方图。该render处理大部分画图工作，我们后续代码也会看到可以使用另一个类型的render替换现有的render，来改变图表的整个外观。
- Dataset、axes和render都由CategorryPlot来管理，CategorryPlot系统组件之间的大部分交互工作。当我们定制一个图表时，我们经常需要先获得整个图表plot、renderer和dataset的引用。在代码的第四行，创建了一个plot，然后其他组件对它进行赋值。
- 最后，在JFreeChart实例中，这个plot用指定的标题被封装。JFreeChart类提供了比较高层次的访问图表。但在这个plot曾思图表就大部分被定义出来了（Plot管理很多对象，例如axes、dataset和renderer）。

图表的内部结构基本上是由上面的知识理论组成。在后续的章节，我们会逐渐学习更多的定制我们图表的方法。

6.4 直方条形图的简单定制

调用JFreeChart和CategoryPlot类方法可以进行一些简单的直方图表外观的修改。例如，改变图表和区域的背景颜色代码如下：

```
chart.setBackgroundPaint(Color.white);
CategoryPlot plot = (CategoryPlot) chart.getPlot();
plot.setBackgroundPaint(Color.lightGray);
plot.setRangeGridlinePaint(Color.white);
```

该片段代码（摘自BarExample2.java类）显示了改变图表的背景颜色、获得图表的plot（区域）的引用，并且进行了修改——效果如图6.3.

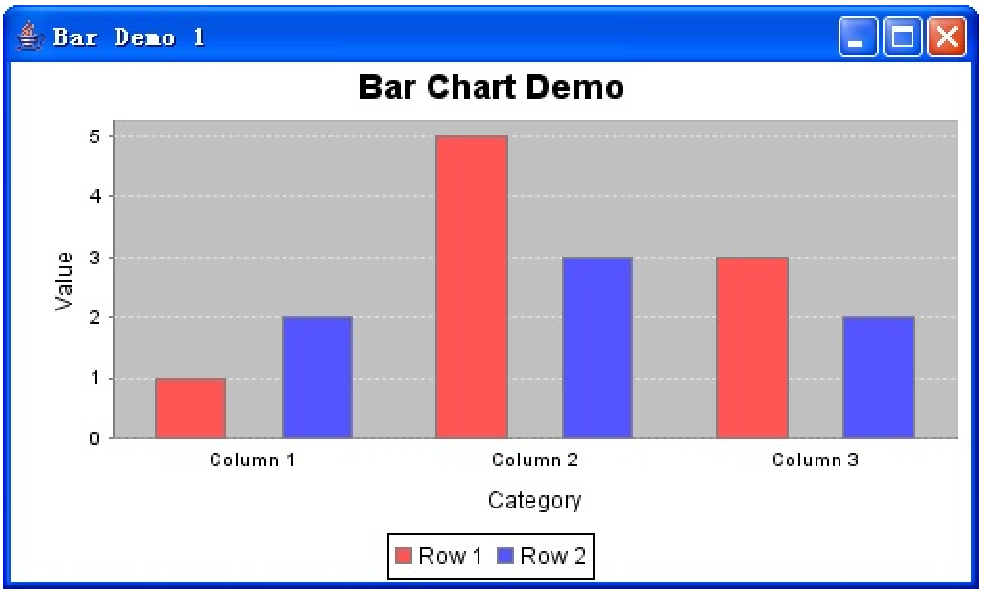


图6.3一个直方条形图（参考：BarExample2.java）

区域Plot的引用（CategoryPlot）是必须的——转换类型也是非常安全的，因为我们知道该图表类型使用CategoryPlot。JFreeChart使用不同的区域类型（比如PiePlot、XYPlot）控制不同类型的图表。我们必须将plot的引用转化成图表响应的类型，因为基本类Plot仅仅定义了一些通用的属性和方法。随着对JFreeChart了解的加深，我们将学习每一种图表使用的不同的plot子类。

在我们的例子中，我们使用plot的引用来改变水平轴的网格线颜色。看一下CategoryPlot类的API文件，就会看到我们能够修改的地方。

6.5 定制外观

回顾6.3节内容，CategoryPlot管理这一个BarRenderer的实例renderer。如果我们想获得这个renderer的引用，大量的定制选择项会变得有效。

6.5.1 直方条形图颜色

改变图表中每个系列直方图的颜色，使用如下代码：

```
BarRenderer renderer = (BarRenderer) plot.getRenderer();
renderer.setSeriesPaint(0, Color.gray);
renderer.setSeriesPaint(1, Color.orange);
renderer.setDrawBarOutline(false);
```

运行上面代码显示的结果如下图6.4.注意setSeriesPaint（）方法是在抽象AbstractRenderer基类里面定义的——所以，我们可以在任何类型的renderer里面使用。

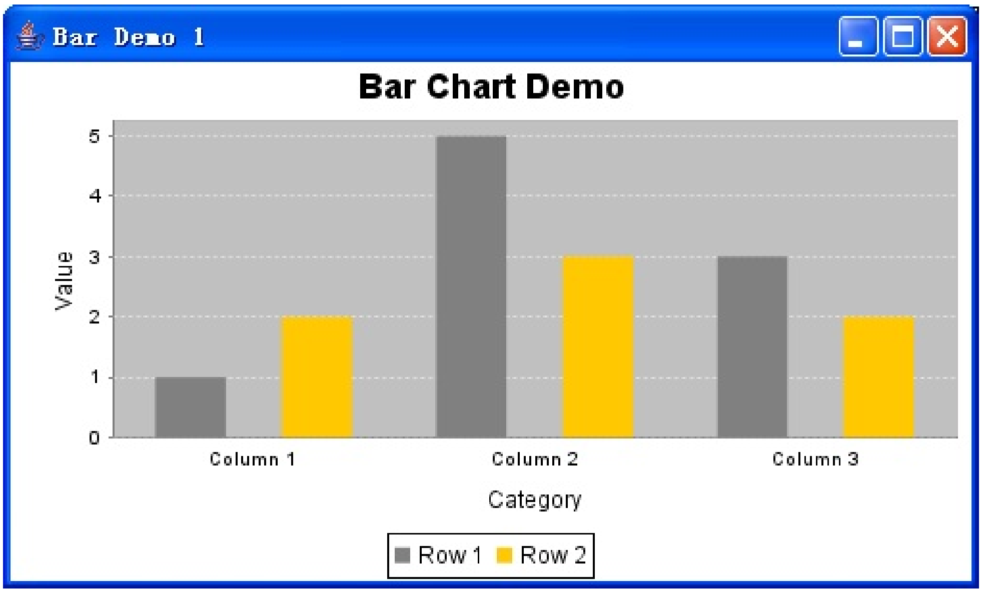


图6.4一个直方条形图（参考：BarExample3.java）

6.5.2 种类里直方条形图之间的空间

此外，renderer还可以控制每个种类中直方条形图之间的间距。因此我们可以在同一个种类中将空间完全去掉，代码如下：

```
BarRenderer renderer = (BarRenderer) plot.getRenderer();
renderer.setItemMargin(0.0);
```

代码显示的结果如图6.5所示。

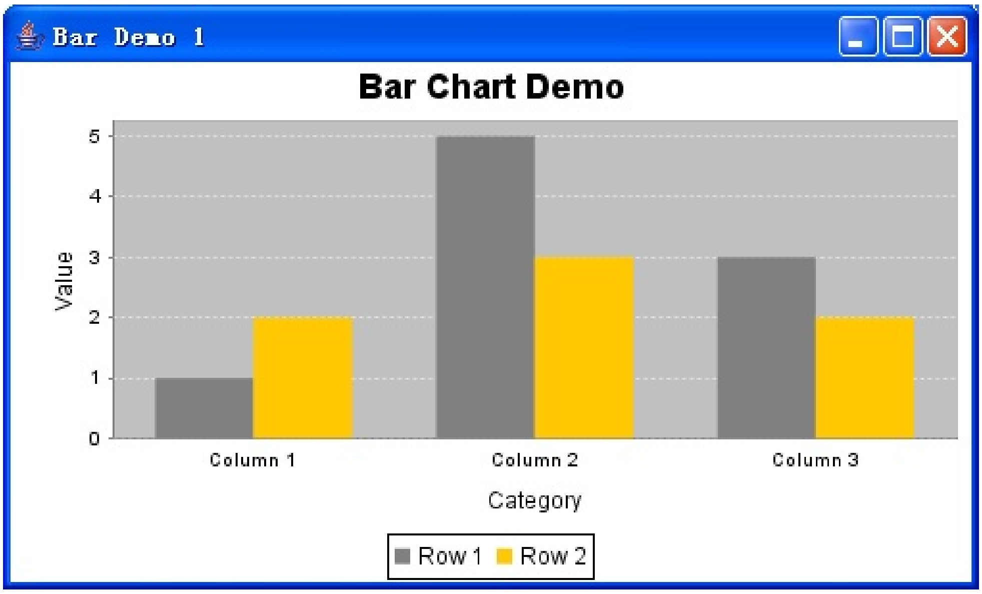


图6.5一个直方条形图（参考：BarExample4.java）

注意条形图看上去有点变宽——主要是因为JFreeChart分配空间时，分配给种类条形图之间的间距的尺度比较少，所以看上去就显得有点长宽了。

6.6 示例代码解读

6.6.1 体会

与饼图的数据集不同之处在于：

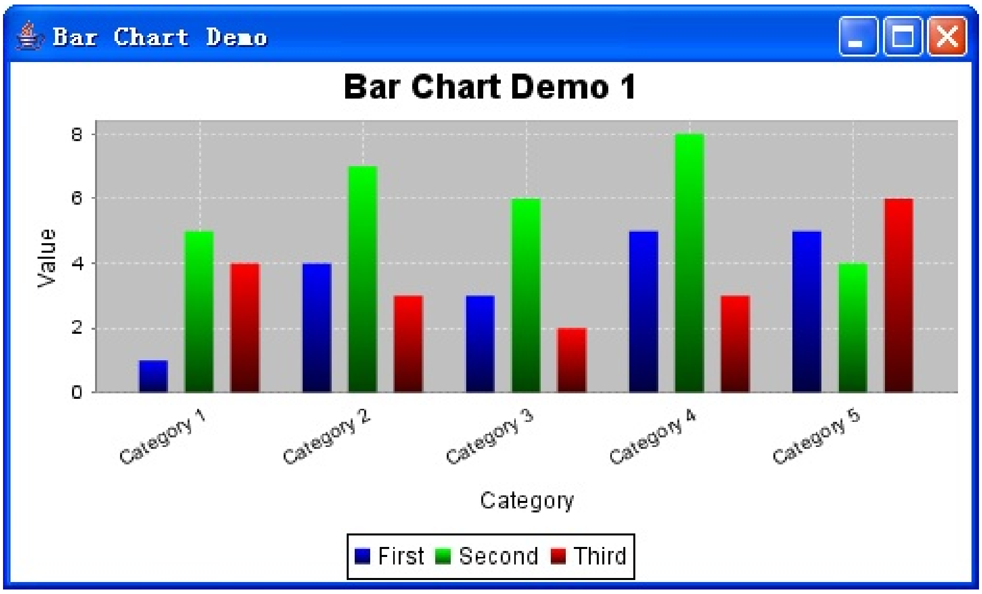
饼图数据集是key/value二维数据（PieDataset）。而直方条形图需要三维数据（CategoryDataset）。

6.6.2 类BarChartDemo1.java

功能：

一个简单的直方条形图。使用GradientPaint实例对象为每一个系列修改renderer

效果：



代码：

```
public class BarChartDemo1 extends ApplicationFrame {
    private static final long serialVersionUID = 1L;
    public BarChartDemo1(String string) {
        super(string);
        JPanel jpanel = createDemoPanel();
        jpanel.setPreferredSize(new Dimension(500, 270));
        setContentPane(jpanel);
    }
    private static CategoryDataset createDataset() {
        String string = "First";
        String string_0_ = "Second";
        String string_1_ = "Third";
        String string_2_ = "Category 1";
```

```
        String string_3_ = "Category 2";
        String string_4_ = "Category 3";
        String string_5_ = "Category 4";
        String string_6_ = "Category 5";
        DefaultCategoryDataset defaultcategorydataset = new Default
        defaultcategorydataset.addValue(1.0, string, string_2_);
        defaultcategorydataset.addValue(4.0, string, string_3_);
        defaultcategorydataset.addValue(3.0, string, string_4_);
        defaultcategorydataset.addValue(5.0, string, string_5_);
        defaultcategorydataset.addValue(5.0, string, string_6_);
        defaultcategorydataset.addValue(5.0, string_0_, string_2_);
        defaultcategorydataset.addValue(7.0, string_0_, string_3_);
        defaultcategorydataset.addValue(6.0, string_0_, string_4_);
        defaultcategorydataset.addValue(8.0, string_0_, string_5_);
        defaultcategorydataset.addValue(4.0, string_0_, string_6_);
        defaultcategorydataset.addValue(4.0, string_1_, string_2_);
        defaultcategorydataset.addValue(3.0, string_1_, string_3_);
        defaultcategorydataset.addValue(2.0, string_1_, string_4_);
        defaultcategorydataset.addValue(3.0, string_1_, string_5_);
        defaultcategorydataset.addValue(6.0, string_1_, string_6_);
        return defaultcategorydataset;
    }
    private static JFreeChart createChart(CategoryDataset category
        JFreeChart jfreechart = ChartFactory.createBarChart("Bar Ch
            "Category", "Value", categorydataset, PlotOrientation.V
            true, true, false);
        jfreechart.setBackgroundPaint(Color.WHITE);
        CategoryPlot categoryplot = (CategoryPlot) jfreechart.getPl
        categoryplot.setBackgroundPaint(Color.lightGray);
        categoryplot.setDomainGridlinePaint(Color.white);
        categoryplot.setDomainGridlinesVisible(true);
        categoryplot.setRangeGridlinePaint(Color.white);
        //刻度轴刻度设置
        NumberAxis numberaxis = (NumberAxis) categoryplot.getRangeA
        numberaxis.setStandardTickUnits(NumberAxis.createIntegerTic
        //renderer设置
        BarRenderer barrenderer = (BarRenderer) categoryplot.getRen
        barrenderer.setDrawBarOutline(false);//设置外廓线不可见
        GradientPaint gradientpaint = new GradientPaint(0.0F, 0.0F,
            0.0F, 0.0F, new Color(0, 0, 64));
        GradientPaint gradientpaint_7_ = new GradientPaint(0.0F, 0.
            Color.green, 0.0F, 0.0F, new Color(0, 64, 0));
        GradientPaint gradientpaint_8_ = new GradientPaint(0.0F, 0.
            Color.red, 0.0F, 0.0F, new Color(64, 0, 0));
        barrenderer.setSeriesPaint(0, gradientpaint);
        barrenderer.setSeriesPaint(1, gradientpaint_7_);
        barrenderer.setSeriesPaint(2, gradientpaint_8_);
        //设置种类标签旋转的角度，逆时针旋转
        CategoryAxis categoryaxis = categoryplot.getDomainAxis();
        categoryaxis.setCategoryLabelPositions(CategoryLabelPositio
            .createUpRotationLabelPositions(Math.PI / 6));
        return jfreechart;
    }
```

```
    public static JPanel createDemoPanel() {
        JFreeChart jfreechart = createChart(createDataset());
        return new ChartPanel(jfreechart);
    }
    public static void main(String[] strings) {
        BarChartDemo1 barchartdemo1 = new BarChartDemo1("Bar Chart
        barchartdemo1.pack();
        RefineryUtilities.centerFrameOnScreen(barchartdemo1);
        barchartdemo1.setVisible(true);
    }
}
```

程序代码说明：

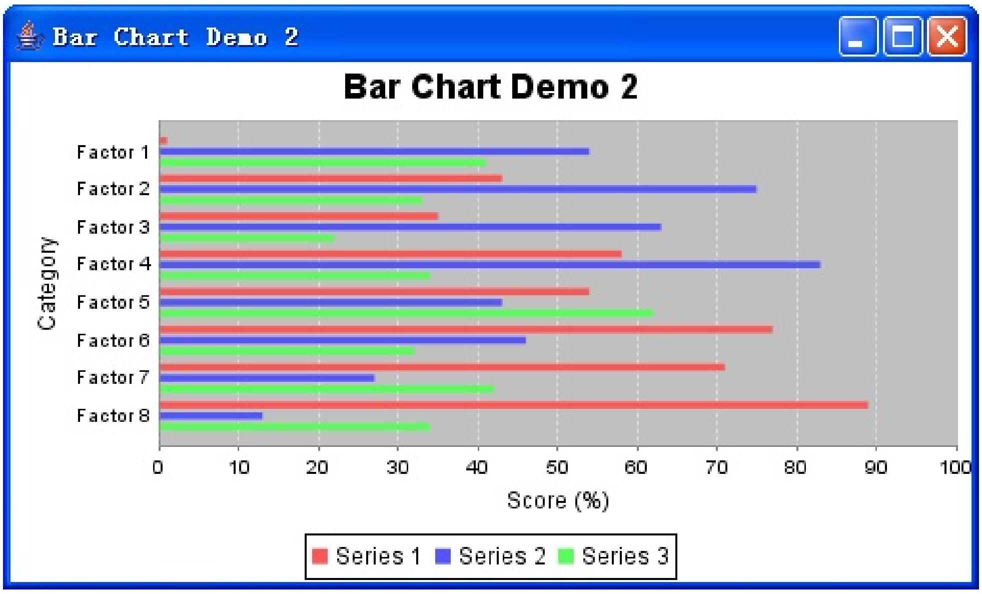
- `Main()` 方法执行直方条形图。编写方法与饼图一样。
- `BarChartDemo1` 构造函数中创了一个JPanel，并设置大小。
- `createDemoPanel()` 方法创建了一个JPanel，并且在该panel上创建了直方条形图。
- `createDataset()` 方法创建了数据集。类型为CategoryDataset。注意数据集为三维数据。与饼图不同。
- 使用 `ChartFactory.createBarChart()` 方法创建直方条形图
- `jfreechart.setBackgroundPaint(Color.WHITE)` ：设置图表的背景颜色。
- `categoryplot.setBackgroundPaint(Color.lightGray)` ：设置直方条形图的背景颜色。
- `setDomainGridlinePaint（Color.whites）` ：设置垂直格线的颜色。默认不可见。
- `setRangeGridlinePaint(Color.white)` ：设置水平格线的颜色。默认可见。
- `setStandardTickUnits(NumberAxis.createIntegerTickUnits())` ：设置数据轴的刻度递进范围。
- `GradientPaint` 类用来设置渐变色。
- `categoryaxis.setCategoryLabelPositions()` ：设置标签文字旋转的角度。

6.6.3 类BarChartDemo2.java

功能：

显示水平的直方条形图。

效果：



代码：

```
private static CategoryDataset createDataset() {
    double[][] ds = { { 1.0, 43.0, 35.0, 58.0, 54.0, 77.0, 71.0, 89
    { 54.0, 75.0, 63.0, 83.0, 43.0, 46.0, 27.0, 13.0 },
    { 41.0, 33.0, 22.0, 34.0, 62.0, 32.0, 42.0, 34.0 } };
    return DatasetUtilities.createCategoryDataset("Series ", "Facto
}
private static JFreeChart createChart(CategoryDataset categorydatas
    JFreeChart jfreechart = ChartFactory.createBarChart("Bar Chart
        "Category", "Score (%)", categorydataset,
    PlotOrientation.VERTICAL, true, true, false);
    jfreechart.setBackgroundPaint(Color.white);
    CategoryPlot categoryplot = (CategoryPlot) jfreechart.getPlot()
    categoryplot.setBackgroundPaint(Color.lightGray);
    categoryplot.setRangeGridlinePaint(Color.white);
    categoryplot.setRangeAxisLocation(AxisLocation.BOTTOM_OR_LEFT);
    NumberAxis numberaxis = (NumberAxis) categoryplot.getRangeAxis(
    numberaxis.setRange(0.0, 100.0);
    numberaxis.setStandardTickUnits(NumberAxis.createIntegerTickUni
    BarRenderer barrenderer = (BarRenderer) categoryplot.getRendere
    barrenderer.setDrawBarOutline(false);
    barrenderer
        .setLegendItemToolTipGenerator(new StandardCategorySeriesLa
        "Tooltip: {0}"));
    return jfreechart;
}
```

程序代码说明：

- 数据集的创建另一种方式，使用二维数组。

7 折线图

7.1 简介

本章讲述了JFreeChart创建折线图的内容。我们可以使用CategoryDataset或XYDataset数据集接口创建折线图。

7.2 使用categoryDataset数据集创建折线图

7.2.1 概述

使用CategoryDataset创建的折线图将每个数据点（种类，值）使用一条直线连接起来。本章讲的一个简单应用产生如下界面，如图7.1：

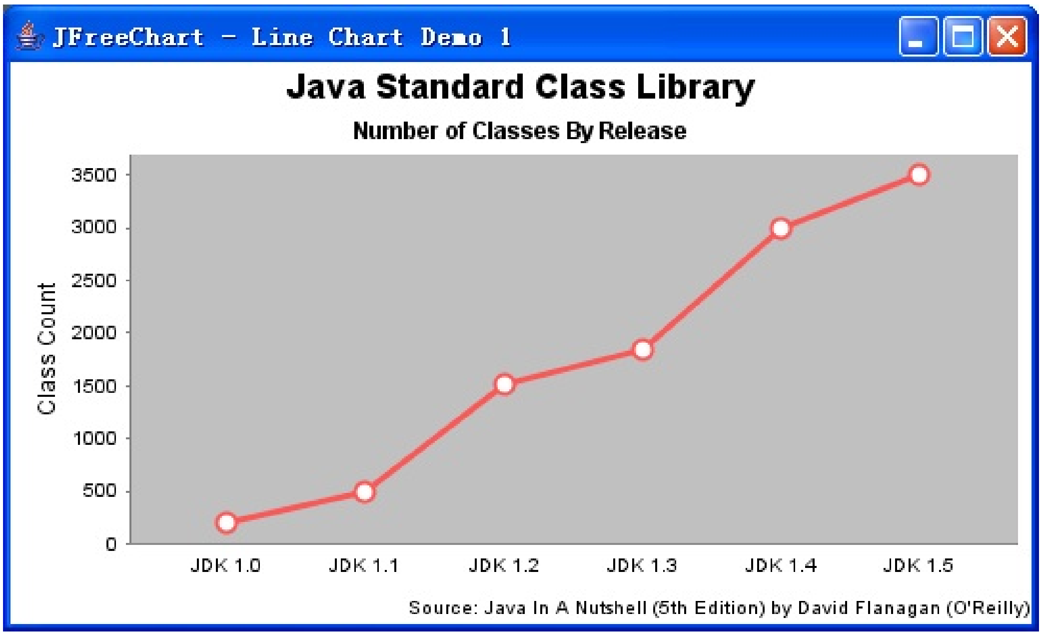


图7.1 一个简单的折线图

全部的代码简JFreeChart开发指南一并下载的demo（参考：LineChartDemo1.java）。

7.2.2 CategoryDataset

正如其他图表一样，创建折线图的第一步是创建第一个dataset。在本例子中，使用DefaultCategoryDataset，代码如下：

```
private static CategoryDataset createDataset() {
    DefaultCategoryDataset defaultcategorydataset = new DefaultCate
    defaultcategorydataset.addValue(212.0, "Classes", "JDK 1.0");
    defaultcategorydataset.addValue(504.0, "Classes", "JDK 1.1");
    defaultcategorydataset.addValue(1520.0, "Classes", "JDK 1.2");
    defaultcategorydataset.addValue(1842.0, "Classes", "JDK 1.3");
    defaultcategorydataset.addValue(2991.0, "Classes", "JDK 1.4");
    defaultcategorydataset.addValue(3500.0, "Classes", "JDK 1.5");
    return defaultcategorydataset;
}
```

注意：你可以使用任何实现Category接口的数据集。

7.2.3 创建图表

ChartFactory类提供了一个便利的方法createLineChart()创建折线图。代码如下：

```
JFreeChart jfreechart = ChartFactory.createLineChart(
    "Java Standard Class Library",// 图表标题
    null, // 主轴标签
    "Class Count",// 范围轴标签
    categorydataset, // 数据集
    PlotOrientation.VERTICAL,// 方向
    false, // 是否包含图例
    true, // 提示信息是否显示
    false// 是否使用urls
);
```

该方法构建了一个带有标题、图例、和相应的数轴和心态提示产生器的JFreeChart对象。创建Dataset数据集的过程见上节。

7.2.4 定制图表

折线图表将使用大部分缺省的属性来进行初始化。当然了，我们也可以随意修改折线图的属性，来改变我们图表的外观。在本例中，我们通过下面的方式定制折线图：

- 在图表上添加两个副标题；
- 图表的背景颜色设成白色；
- 图区背景颜色设成亮灰色；
- 网格线颜色改变成白色；
- 范围轴修改成仅显示整数数值；
- renderer使用白色填充的形状。

首先，将副标题添加在缺省的位置（主标题下方），代码如下：

```
jfreechart.addSubtitle(new TextTitle("Number of Classes By Release'
```

第二个副标题加入了一些额外的代码，来改变字体，并放置在图表的下方，并且靠右对其，代码如下：

```
TextTitle texttitle = (new TextTitle("Source: Java In A Nutshell (5
texttitle.setFont(new Font("SansSerif", 0, 10));
texttitle.setPosition(RectangleEdge.BOTTOM);
texttitle.setHorizontalAlignment(HorizontalAlignment.RIGHT);
jfreechart.addSubtitle(texttitle);
```

改变图表的背景颜色非常简单，因为JFreeChart类就有设置背景颜色的属性。代码如下：

```
//改变图表的背景颜色
jfreechart.setBackgroundPaint(Color.white);
```

如果改变其他属性的，则需要首先获得图表CategoryPlot对象的引用，然后对该引用进行相应属性的设置。获得对象引用的代码如下：

```
CategoryPlot categoryplot = (CategoryPlot) jfreechart.getPlot();
```

使用CategoryPlot设置图区的背景颜色为亮灰色，设置网格线颜色为白色的代码如下：

```
categoryplot.setBackgroundPaint(Color.lightGray);
categoryplot.setRangeGridlinePaint(Color.white);
```

图区负责在图表上画出数据和轴。其中一部分工作由renderer来完成，我们可以通过getRender()来获得一个renderer。renderer维护大部分与图表内数据项的显示相关的属性。代码如下：

```
LineAndShapeRenderer renderer = (LineAndShapeRenderer) categoryplot
renderer.setShapesVisible(true);
renderer.setDrawOutlines(true);
renderer.setUseFillPaint(true);
```

同时图区也管理着图表所有的轴。在本实例中，修改范围轴以便范围轴的刻度标签显示为整数值。

```
NumberAxis rangeAxis = (NumberAxis) categoryplot.getRangeAxis();
rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits()
```

定制图表还有很多其他的方式。具体的内容详见本文档相关章节，API文档以及源代码实例。

7.2.5 程序的全部代码

```
/* -------------------
* LineChartDemo1.java
* -------------------
* (C) Copyright 2002-2005, by Object Refinery Limited.
*
*/
package demo;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.geom.Ellipse2D;
import java.net.URL;
import javax.swing.ImageIcon;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.LineAndShapeRenderer;
import org.jfree.chart.title.TextTitle;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.HorizontalAlignment;
import org.jfree.ui.RectangleEdge;
import org.jfree.ui.RefineryUtilities;
public class LineChartDemo1 extends ApplicationFrame {
    /**
    *
    */
    private static final long serialVersionUID = -63543506043130797
    /* synthetic */static Class class$demo$LineChartDemo1;
    public LineChartDemo1(String string) {
        super(string);
        JPanel jpanel = createDemoPanel();
        jpanel.setPreferredSize(new Dimension(500, 270));
        setContentPane(jpanel);
    }
    private static CategoryDataset createDataset() {
        DefaultCategoryDataset defaultcategorydataset = new Default
        defaultcategorydataset.addValue(212.0, "Classes", "JDK 1.0"
        defaultcategorydataset.addValue(504.0, "Classes", "JDK 1.1"
        defaultcategorydataset.addValue(1520.0, "Classes", "JDK 1.2
        defaultcategorydataset.addValue(1842.0, "Classes", "JDK 1.3
        defaultcategorydataset.addValue(2991.0, "Classes", "JDK 1.4
        defaultcategorydataset.addValue(3500.0, "Classes", "JDK 1.5
        return defaultcategorydataset;
```

```
    }
    private static JFreeChart createChart(CategoryDataset categoryc
        JFreeChart jfreechart = ChartFactory.createLineChart(
            "Java Standard Class Library",// 图表标题
            null, // 主轴标签
            "Class Count",// 范围轴标签
            categorydataset, // 数据集
            PlotOrientation.VERTICAL,// 方向
            false, // 是否包含图例
            true, // 提示信息是否显示
            false// 是否使用urls
        );
        // 添加主标题
        jfreechart.addSubtitle(new TextTitle("Number of Classes By 
        TextTitle texttitle = (new TextTitle(
            "Source: Java In A Nutshell (5th Edition) by David Flar
        texttitle.setFont(new Font("SansSerif", 0, 10));
        texttitle.setPosition(RectangleEdge.BOTTOM);
        texttitle.setHorizontalAlignment(HorizontalAlignment.RIGHT)
        jfreechart.addSubtitle(texttitle);
        // 改变图表的背景颜色
        jfreechart.setBackgroundPaint(Color.white);
        CategoryPlot categoryplot = (CategoryPlot) jfreechart.getPl
        categoryplot.setBackgroundPaint(Color.lightGray);
        categoryplot.setRangeGridlinePaint(Color.white);
        categoryplot.setRangeGridlinesVisible(false);
        URL url = (class$demo$LineChartDemo1 == null ? class$demo$L
            : class$demo$LineChartDemo1).getClassLoader().getResour
            "OnBridge11small.png");
        if (url != null) {
            ImageIcon imageicon = new ImageIcon(url);
            jfreechart.setBackgroundImage(imageicon.getImage());
            categoryplot.setBackgroundPaint(new Color(0, 0, 0, 0));
        }
        NumberAxis numberaxis = (NumberAxis) categoryplot.getRangeA
        numberaxis.setStandardTickUnits(NumberAxis.createIntegerTic
        LineAndShapeRenderer lineandshaperenderer = (LineAndShapeRe
            .getRenderer();
        lineandshaperenderer.setShapesVisible(true);
        lineandshaperenderer.setDrawOutlines(true);
        lineandshaperenderer.setUseFillPaint(true);
        lineandshaperenderer.setBaseFillPaint(Color.white);
        lineandshaperenderer.setSeriesStroke(0, new BasicStroke(3.0
        lineandshaperenderer.setSeriesOutlineStroke(0, new BasicStr
        lineandshaperenderer.setSeriesShape(0, new Ellipse2D.Double
            10.0, 10.0));
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) cate
            .getRenderer();
        renderer.setShapesVisible(true);
        renderer.setDrawOutlines(true);
        renderer.setUseFillPaint(true);
        return jfreechart;
    }
```

```
    public static JPanel createDemoPanel() {
        JFreeChart jfreechart = createChart(createDataset());
        return new ChartPanel(jfreechart);
    }
    public static void main(String[] strings) {
        LineChartDemo1 linechartdemo1 = new LineChartDemo1(
        "JFreeChart - Line Chart Demo 1");
        linechartdemo1.pack();
        RefineryUtilities.centerFrameOnScreen(linechartdemo1);
        linechartdemo1.setVisible(true);
    }
    /* synthetic */
    static Class class$(String string) {
        Class var_class;
        try {
            var_class = Class.forName(string);
        } catch (ClassNotFoundException classnotfoundexception) {
            throw new NoClassDefFoundError(classnotfoundexception.g
        }
        return var_class;
    }
}
```

7.3 使用XYDataset数据集创建折线图

7.3.1 概述

折线图也可以使用XYDataset数据集，使用一条直线将相邻的点（x，y）点连接起来。本章介绍的一个使用XYDataset数据集创建折线图的简单实例，如下图7.2所示。

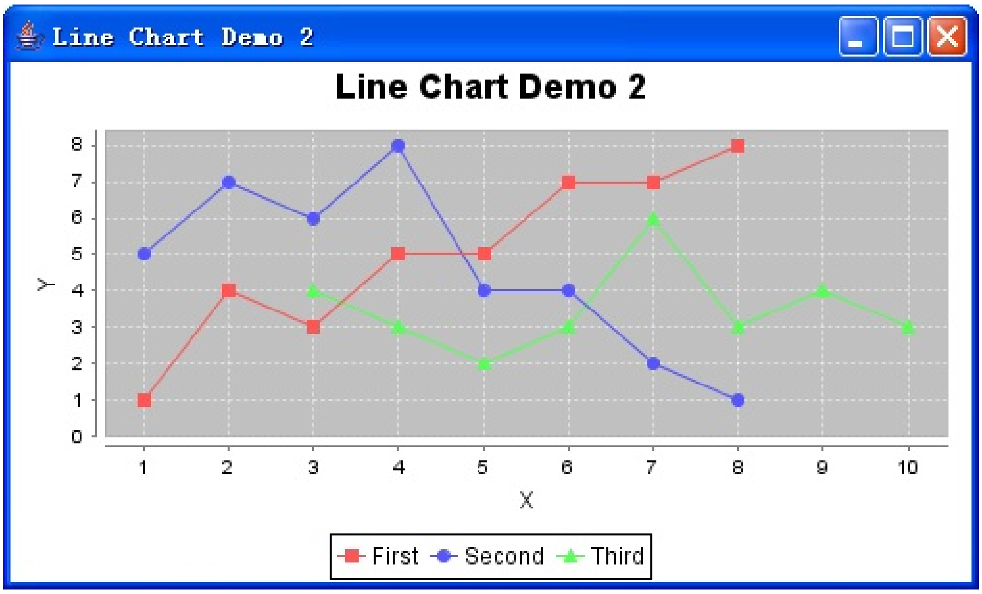


图7.2 一个简单的基于XYDataset数据集的折线图（参考：LineChartDemo2.java）

7.3.2 XYDataset

对于该图表来说，使用的数据集是XYSeriesCollection（当然我们可以使用实现XYDataset接口的其他数据集）。出于独立演示的目点，我们创建的dataset代码如下：

```
private static XYDataset createDataset() {
    XYSeries xyseries = new XYSeries("First");
    xyseries.add(1.0, 1.0);
    xyseries.add(2.0, 4.0);
    xyseries.add(3.0, 3.0);
    xyseries.add(4.0, 5.0);
    xyseries.add(5.0, 5.0);
    xyseries.add(6.0, 7.0);
    xyseries.add(7.0, 7.0);
    xyseries.add(8.0, 8.0);
    XYSeries xyseries_0_ = new XYSeries("Second");
    xyseries_0_.add(1.0, 5.0);
    xyseries_0_.add(2.0, 7.0);
    xyseries_0_.add(3.0, 6.0);
    xyseries_0_.add(4.0, 8.0);
    xyseries_0_.add(5.0, 4.0);
    xyseries_0_.add(6.0, 4.0);
    xyseries_0_.add(7.0, 2.0);
    xyseries_0_.add(8.0, 1.0);
    XYSeries xyseries_1_ = new XYSeries("Third");
    xyseries_1_.add(3.0, 4.0);
    xyseries_1_.add(4.0, 3.0);
    xyseries_1_.add(5.0, 2.0);
    xyseries_1_.add(6.0, 3.0);
    xyseries_1_.add(7.0, 6.0);
    xyseries_1_.add(8.0, 3.0);
    xyseries_1_.add(9.0, 4.0);
    xyseries_1_.add(10.0, 3.0);
    XYSeriesCollection xyseriescollection = new XYSeriesCollection(
    xyseriescollection.addSeries(xyseries);
    xyseriescollection.addSeries(xyseries_0_);
    xyseriescollection.addSeries(xyseries_1_);
    return xyseriescollection;
}
```

注意：每个系列必须有x值（不是必须有y值），并且该系列独立于其他系列。数据集可以接受一个y值为null的值。当图表遇到null值时，连接线不被画出，该系列的连线不会连续。出现下图7.3类型。

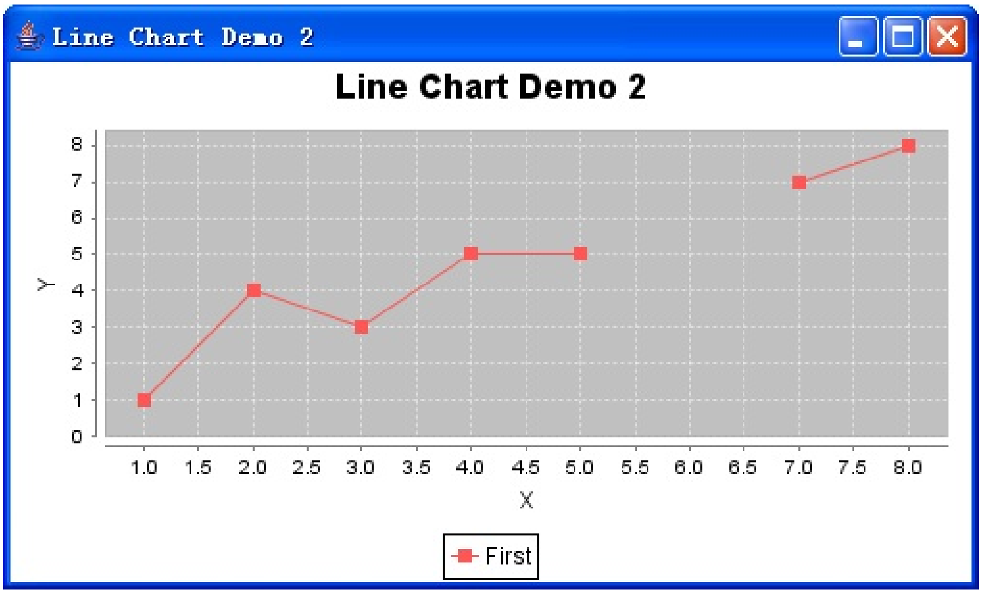


图7.3有一个 y值为null时，图表显示断续。

7.3.3 创建图表

ChartFactory类提供了一个便利的方法createXYLineChart()创图表：

```
JFreeChart jfreechart = ChartFactory.createXYLineChart(
    "Line Chart Demo 2", // chart title
    "X", // x axis label
    "Y", // y axis label
    xydataset, // data
    PlotOrientation.VERTICAL,
    true, // include legend
    true, // tooltips
    false // urls
);
```

上面方法构建了一个JFreeChart对象，该对象具有一个标题、图例和相关轴的图区及renderer。数据集使用上节我们创建的数据集。

7.3.4 定制图表

图表将使用大部分缺省的属性进行初始化设置。当然了，我们也可以随意修改这些属性，来改变我们图表的外观。在本实例中，设置的几个属性如下：

- 设置图表的背景颜色
- 设置图区的背景颜色
- 设置轴的平移值
- 设置主轴和范围轴网格线颜色
- 修改renderer改变连线点的形状
- 范围轴刻度的设置，以便显示整数值。

改变图表背景颜色非常简单。代码如下：

```
//改变图表的背景颜色
jfreechart.setBackgroundPaint(Color.white);
```

改变图区背景颜色、轴平移、网格线颜色，需要使用plot图区对象的一个引用来修改。图片对象需要转化成XYPlot对象，主要是因为我们可以访问更多更具体的图区方法。代码如下：

```
XYPlot xyplot = (XYPlot) jfreechart.getPlot();
xyplot.setBackgroundPaint(Color.lightGray);
xyplot.setAxisOffset(new RectangleInsets(5.0, 5.0, 5.0, 5.0));
xyplot.setDomainGridlinePaint(Color.white);
xyplot.setRangeGridlinePaint(Color.white);
```

修改renderer来显示连线之间的形状。代码如下：

```
XYLineAndShapeRenderer xylineandshaperenderer = (XYLineAndShapeRenc
xylineandshaperenderer.setShapesVisible(true);
xylineandshaperenderer.setShapesFilled(true);
```

最后就是修改范围轴。我们将默认刻度值（允许显示小数）改成只显示整数的刻度值。代码如下：

```
NumberAxis numberaxis = (NumberAxis) xyplot.getRangeAxis();
numberaxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits()
```

参考源代码、Javadoc的API文档以及其他相关XYPlot的定制内容，来学习更多的细节。

8 时序图

8.1 简介

时序图类似于折线图，唯一不同的地方是时序图的主轴是日期而不是数值。本章讲述如何使用JFreeChart创建时序图。

8.2 创建时序图

8.2.1 概述

时序图表的确是一个使用XYDataset数据集的折线图。不同点就是再主轴上X轴值显示的是日期。本章讲述的一个简单应用如下图8.1所示

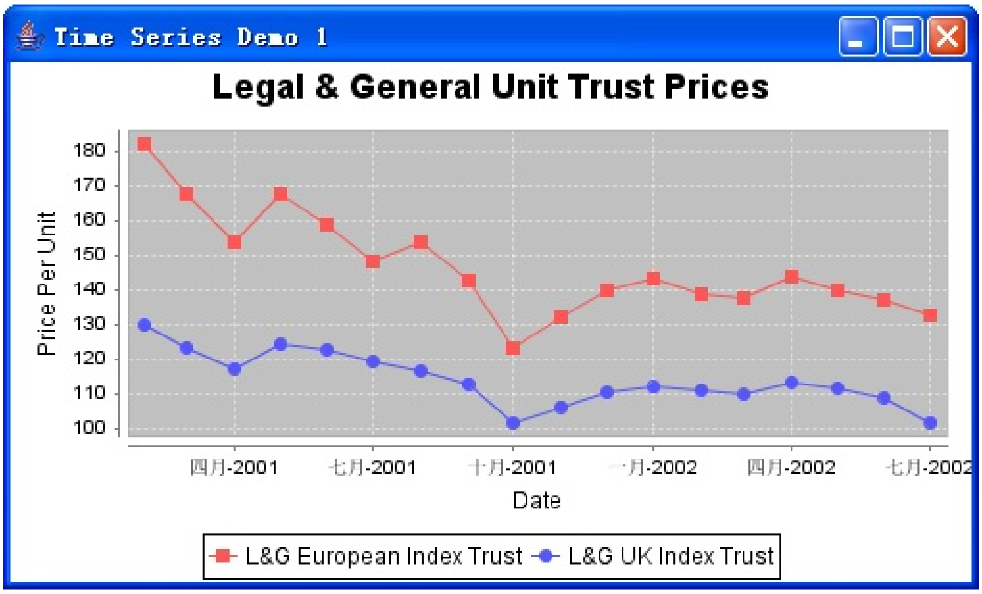


图8.1 一个简单的时序图表（参考：TimeSeriesDemo1.java）。

上图实例代码参见类TimeSeriesDemo1.java。

8.2.2 日期还是数字？

创建序图使用的数据集是XYDataset。接口不能有返回日期类型以外的方法。那么JFreeChart是如何创建时序图表的呢？

数据集返回的x值是基本的double类型，但这个值通过一种特殊的方式来进行转译成日期——该数是反映了一个从1970/1/1起计算的一个毫秒级值（译码过程使用java.util.Data类计算）。

具体的轴类（DateAxis）将毫秒级数据转化成日期，并作为需要返回该值，将数值作为主轴刻度显示出来。

8.2.3 数据集

演示的本实例，数据使用的是一个TimeSeriesCollection对象（我们可以是任何XYDataset接口的实现）。代码如下：

```
private static XYDataset createDataset() {
    TimeSeries timeseries = new TimeSeries(
        "L&G European Index Trust",
        (class$org$jfree$data$time$Month == null ? (class$org$jfree
        : class$org$jfree$data$time$Month));
    timeseries.add(new Month(2, 2001), 181.8);
    timeseries.add(new Month(3, 2001), 167.3);
    timeseries.add(new Month(4, 2001), 153.8);
    timeseries.add(new Month(5, 2001), 167.6);
    timeseries.add(new Month(6, 2001), 158.8);
    timeseries.add(new Month(7, 2001), 148.3);
    timeseries.add(new Month(8, 2001), 153.9);
    timeseries.add(new Month(9, 2001), 142.7);
    timeseries.add(new Month(10, 2001), 123.2);
    timeseries.add(new Month(11, 2001), 131.8);
    timeseries.add(new Month(12, 2001), 139.6);
    timeseries.add(new Month(1, 2002), 142.9);
    timeseries.add(new Month(2, 2002), 138.7);
    timeseries.add(new Month(3, 2002), 137.3);
    timeseries.add(new Month(4, 2002), 143.9);
    timeseries.add(new Month(5, 2002), 139.8);
    timeseries.add(new Month(6, 2002), 137.0);
    timeseries.add(new Month(7, 2002), 132.8);
    TimeSeries timeseries_0_ = new TimeSeries(
        "L&G UK Index Trust",
        (class$org$jfree$data$time$Month == null ? (class$org$jfree
        : class$org$jfree$data$time$Month));
    timeseries_0_.add(new Month(2, 2001), 129.6);
    timeseries_0_.add(new Month(3, 2001), 123.2);
    timeseries_0_.add(new Month(4, 2001), 117.2);
    timeseries_0_.add(new Month(5, 2001), 124.1);
    timeseries_0_.add(new Month(6, 2001), 122.6);
    timeseries_0_.add(new Month(7, 2001), 119.2);
    timeseries_0_.add(new Month(8, 2001), 116.5);
    timeseries_0_.add(new Month(9, 2001), 112.7);
    timeseries_0_.add(new Month(10, 2001), 101.5);
    timeseries_0_.add(new Month(11, 2001), 106.1);
    timeseries_0_.add(new Month(12, 2001), 110.3);
    timeseries_0_.add(new Month(1, 2002), 111.7);
    timeseries_0_.add(new Month(2, 2002), 111.0);
    timeseries_0_.add(new Month(3, 2002), 109.6);
    timeseries_0_.add(new Month(4, 2002), 113.2);
    timeseries_0_.add(new Month(5, 2002), 111.6);
    timeseries_0_.add(new Month(6, 2002), 108.8);
    timeseries_0_.add(new Month(7, 2002), 101.6);
    TimeSeriesCollection timeseriescollection = new TimeSeriesColle
    timeseriescollection.addSeries(timeseries);
    timeseriescollection.addSeries(timeseries_0_);
    return timeseriescollection;
}
```

实例中，系列包含了每月的数据。尽管如此，仍然使用TimeSeries类来显示间隔的时间值（年、日、小时等）。

8.2.4 构建图表

使用ChartFactory类提供的便利方法createTimeSeriesChart()创建图表，代码如下：

```
JFreeChart jfreechart = ChartFactory.createTimeSeriesChart(
    "Legal & General Unit Trust Prices", // title
    "Date", // x-axis label
    "Price Per Unit", // y-axis label
    xydataset, // data
    true, // create legend?
    true, // generate tooltips?
    false // generate URLs?
);
```

该方法构建了一个带有标题、图例、相应轴的区域和展示器的JFreeChart对象。使用的数据集见上一节内容。

8.2.5 定制图表

图表的大部分属性使用了缺省的值进行初始化。当然，我们可以随时修改这些属性的设置来改变我们图表的外观展现。在本实例中，修改的几个属性如下：

- 修改renderer，改变每个数据点显示的系列形状，数据点之间的折线除外。
- 主轴的数据格式进行格式化后显示。

修改renderer需要一下两个步骤：获得renderer引用和将renderer对象转化成XYLineAndShapeRenderer类型。代码如下：

```
XYItemRenderer xyitemrenderer = xyplot.getRenderer();
if (xyitemrenderer instanceof XYLineAndShapeRenderer) {
    XYLineAndShapeRenderer xylineandshaperenderer = (XYLineAndShape
    xylineandshaperenderer.setBaseShapesVisible(true);
    xylineandshaperenderer.setBaseShapesFilled(true);
}
```

最后，将格式化的数据传给主轴，以改变显示。代码如下：

```
DateAxis dateaxis = (DateAxis) xyplot.getDomainAxis();
dateaxis.setDateFormatOverride(new SimpleDateFormat("MMM-yyyy"));
```

当设置了Dataaxis时，系统将自动选择一个DataTickUnit来显示主轴刻度。但系统将使用上面我们格式化的数据来显示，而不是系统默认的格式。

8.2.6 全部代码

文档的全部代码如下：

```
/* TimeSeriesDemo1 - Decompiled by JODE
* Visit http://jode.sourceforge.net/
*/
package demo;
import java.awt.Color;
import java.awt.Dimension;
import java.text.SimpleDateFormat;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.DateAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.time.Month;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.xy.XYDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RectangleInsets;
import org.jfree.ui.RefineryUtilities;
public class TimeSeriesDemo1 extends ApplicationFrame {
    private static final long serialVersionUID = -54122863709566463
    /* synthetic */
    static Class class$org$jfree$data$time$Month;
    public TimeSeriesDemo1(String string) {
        super(string);
        XYDataset xydataset = createDataset();
        JFreeChart jfreechart = createChart(xydataset);
        ChartPanel chartpanel = new ChartPanel(jfreechart, false);
        chartpanel.setPreferredSize(new Dimension(500, 270));
        chartpanel.setMouseZoomable(true, false);
        setContentPane(chartpanel);
    }
    private static JFreeChart createChart(XYDataset xydataset) {
        JFreeChart jfreechart = ChartFactory.createTimeSeriesChart(
            "Legal & General Unit Trust Prices", // title
            "Date", // x-axis label
            "Price Per Unit", // y-axis label
            xydataset, // data
            true, // create legend?
            true, // generate tooltips?
            false // generate URLs?
        );
```

```
        jfreechart.setBackgroundPaint(Color.white);
        XYPlot xyplot = (XYPlot) jfreechart.getPlot();
        xyplot.setBackgroundPaint(Color.lightGray);
        xyplot.setDomainGridlinePaint(Color.white);
        xyplot.setRangeGridlinePaint(Color.white);
        xyplot.setAxisOffset(new RectangleInsets(5.0, 5.0, 5.0, 5.(
        xyplot.setDomainCrosshairVisible(true);
        xyplot.setRangeCrosshairVisible(true);
        org.jfree.chart.renderer.xy.XYItemRenderer xyitemrenderer =
            .getRenderer();
        if (xyitemrenderer instanceof XYLineAndShapeRenderer) {
            XYLineAndShapeRenderer xylineandshaperenderer = (XYLine
            xylineandshaperenderer.setBaseShapesVisible(true);
            xylineandshaperenderer.setBaseShapesFilled(true);
        }
        DateAxis dateaxis = (DateAxis) xyplot.getDomainAxis();
        dateaxis.setDateFormatOverride(new SimpleDateFormat("MMM-yy
        return jfreechart;
    }
    private static XYDataset createDataset() {
        TimeSeries timeseries = new TimeSeries(
            "L&G European Index Trust",
            (class$org$jfree$data$time$Month == null ? (class$org$j
            : class$org$jfree$data$time$Month));
        timeseries.add(new Month(2, 2001), 181.8);
        timeseries.add(new Month(3, 2001), 167.3);
        timeseries.add(new Month(4, 2001), 153.8);
        timeseries.add(new Month(5, 2001), 167.6);
        timeseries.add(new Month(6, 2001), 158.8);
        timeseries.add(new Month(7, 2001), 148.3);
        timeseries.add(new Month(8, 2001), 153.9);
        timeseries.add(new Month(9, 2001), 142.7);
        timeseries.add(new Month(10, 2001), 123.2);
        timeseries.add(new Month(11, 2001), 131.8);
        timeseries.add(new Month(12, 2001), 139.6);
        timeseries.add(new Month(1, 2002), 142.9);
        timeseries.add(new Month(2, 2002), 138.7);
        timeseries.add(new Month(3, 2002), 137.3);
        timeseries.add(new Month(4, 2002), 143.9);
        timeseries.add(new Month(5, 2002), 139.8);
        timeseries.add(new Month(6, 2002), 137.0);
        timeseries.add(new Month(7, 2002), 132.8);
        TimeSeries timeseries_0_ = new TimeSeries(
            "L&G UK Index Trust",
            (class$org$jfree$data$time$Month == null ? (class$org$j
            : class$org$jfree$data$time$Month));
        timeseries_0_.add(new Month(2, 2001), 129.6);
        timeseries_0_.add(new Month(3, 2001), 123.2);
        timeseries_0_.add(new Month(4, 2001), 117.2);
        timeseries_0_.add(new Month(5, 2001), 124.1);
        timeseries_0_.add(new Month(6, 2001), 122.6);
        timeseries_0_.add(new Month(7, 2001), 119.2);
        timeseries_0_.add(new Month(8, 2001), 116.5);
```

```
        timeseries_0_.add(new Month(9, 2001), 112.7);
        timeseries_0_.add(new Month(10, 2001), 101.5);
        timeseries_0_.add(new Month(11, 2001), 106.1);
        timeseries_0_.add(new Month(12, 2001), 110.3);
        timeseries_0_.add(new Month(1, 2002), 111.7);
        timeseries_0_.add(new Month(2, 2002), 111.0);
        timeseries_0_.add(new Month(3, 2002), 109.6);
        timeseries_0_.add(new Month(4, 2002), 113.2);
        timeseries_0_.add(new Month(5, 2002), 111.6);
        timeseries_0_.add(new Month(6, 2002), 108.8);
        timeseries_0_.add(new Month(7, 2002), 101.6);
        TimeSeriesCollection timeseriescollection = new TimeSeries(
        timeseriescollection.addSeries(timeseries);
        timeseriescollection.addSeries(timeseries_0_);
        return timeseriescollection;
    }
    public static JPanel createDemoPanel() {
        JFreeChart jfreechart = createChart(createDataset());
        return new ChartPanel(jfreechart);
    }
    public static void main$(String[] strings) {
        TimeSeriesDemo1 timeseriesdemo1 = new TimeSeriesDemo1(
            "Time Series Demo 1");
        timeseriesdemo1.pack();
        RefineryUtilities.centerFrameOnScreen(timeseriesdemo1);
        timeseriesdemo1.setVisible(true);
    }
    public static void main(String[] args) {
        main$(args);
    }
    /* synthetic */
    static Class class$(String string) {
        Class var_class;
        try {
            var_class = Class.forName(string);
        } catch (ClassNotFoundException classnotfoundexception) {
            throw new NoClassDefFoundError(classnotfoundexception.g
        }
        return var_class;
    }
}
```

9 定制图表（Customising Charts）

9.1 简介

JFreeChart的设计的定制功能是非常灵活的。我们可以使用非常多的属性来设置我们图表的外观。本章将详细介绍一些图表通用的定制技术。

9.2 图表属性

9.2.1 概述

我们可以使用JFreeChart类方法从更高的层次来定制我们图表的外观。可控制的属性有：

- 图表的边框
- 图表的标题和副标题
- 图表的背景颜色和图片
- 使用绘制建议（Rendering Hints）画图表，该属性有是否反锯齿功能。

在下面的章节中将详细描述这些内容。

9.2.2 图表边框

JFreeChart可以在图表的外围画出一个边框。默认状态下，JFreeChart是不画出边框的，但我们可以使用方法setBorderVisible()来设置。边框的颜色和线条风格可使用方法setBorderPaint()和setBorderStroke()来控制。

注意：如果我们在一个ChartPanel里面显示我们的图表，那么我们可能更愿意使用Swing提供的边框。

9.2.3 图表标题

图表有一个标题，显示在图表的顶部、底部、左侧或右侧（同时，我们也可以添加副标题，见下章讲述）。标题使用一个TextTitle的实例对象。我们可以使用getTitle()方法来获得标题的引用。

```
TextTitle texttitle = jfreechart.getTitle();
```

修改标题文本（不修改字体和位置）的代码如下：

```
texttitle.setText("Pie Chart Demo");
```

题头放置在图表的顶部、底部、左侧或右侧的设置，使用标题本书属性设置来完成。下面代码显示的是将标题移植到图表的底部。

```
texttitle.setPosition(RectangleEdge.BOTTOM);
```

如果在我们图表上，我们不希望显示标题，则将标题设置为null即可。

9.2.4 副标题

图表可以拥有任何数量的副标题。添加副标题，需要先创建一个副标题对象（任何Title类的子类），然后将该对象加到图表上即可。代码如下：

```
TextTitle subtitle1 = new TextTitle("A Subtitle");
jfreechart.addSubtitle(subtitle1);
```

我们可以在图表上添加任何数量的副标题，但是紧急我们添加的副标题越多，图表画图的区域就越小。

修改一个已有的副标题，我们需要先获得副标题的一个引用。代码如下：

```
Title subtitle = jfreechart.getSubtitle(0);
```

在我们改变副标题属性之前，我们需要将Title的引用转换成我们需要的适当的子类类型。

我们可以使用getSubtitleCount()方法获得副标题的数量。

9.2.5 设置图表背景颜色

我们可以使用setBackgroundPaint()方法设置图表的背景颜色（注意，我们也可以设置我们图区的背景颜色，这与图表的背景颜色不同）。例如：

```
jfreechart.setBackgroundPaint(Color.blue);
```

我们可使用Paint接口的任何实现作为背景颜色的设置参数，其中有Color、GradientPaint（渐变颜色）和TexturePaint等。代码如下：

```
Paint p = new GradientPaint(0, 0, Color.white, 1000, 0, Color.greer
jfreechart.setBackgroundPaint(p);
```

我们可以设置我们的背景颜色为null，这时推荐使用一个背景图片来设置我们的图表。

9.2.6 使用背景图片

我们可以使用方法setBackgroundImage()来为我们的图表设置一幅背景图表。

```
jfreechart.setBackgroundImage(JFreeChart.INFO.getLogo());
```

默认的，图片充满图表的整个背景，图片失真。但我们可以使用setBackgroundImageAlignment()方法来改变图片不充满整个背景。代码如下：

```
jfreechart.setBackgroundImageAlignment(Align.TOP_LEFT);
```

使用setBackgroundImageAlpha()方法，我们可以控制图片的透明度。如果我们希望图片只填充我们图表的区域（区域包含轴），那么我们需要将背景图片添加到图表的图区。代码如下（以饼图为例）：

```
PiePlot pieplot = (PiePlot) jfreechart.getPlot();
pieplot.setBackgroundImage(JFreeChart.INFO.getLogo())
```

9.2.7 Rendering Hints（绘制建议）

JFreeChart使用java2D的API来画图表。在java2D中的API中，我们可以提供绘制建议让绘制引擎绘制图表。JFreeChart允许我们在画图表时，使用setRenderingHints()方法，将绘制建议参数传入java2D的API中。

JFreeChart还提供了一个便利反锯齿开关方法。当反锯齿开关开时，图表会绘制出比较光滑的图表，但是花费的时间要长。代码如下：

```
jfreechart.setAntiAlias(true);
```

JFreeChart画图时，默认为反锯齿开关为开。

9.3 图区属性

9.3.1 概述

JFreeChart类在绘制图表时，将大部分工作交给了Plot类（图形绘制结构）或Plot的子类。JFreeChart类的getPlot()方法返回了一个图表创建的图区（plot）的引用。

```
Plot plot = jfreechart.getPlot();
```

我们需要将该引用转化成Plot的一个具体子类。例如：

```
CategoryPlot plot = jfreechart.getCategoryPlot();
```

或

```
XYPlot plot = jfreechart.getXYPlot();
```

注意：如果plot不是相应的类，则在转化的时候，会抛出ClassCastException类型转制异常。

9.3.2 图区子类

那么我们如何知道我们图表使用的Plot是那个子类呢？作为使用JFreeChart的经验，分清那些图表使用CategoryPlot和那些图表使用XYPlot是非常清晰的。如果还怀疑，看一下ChartFactory类的源代码就会明白每个类型的图表是如何放在一起的。

9.3.3 设置图区背景颜色

我们可以使用方法setBackgroundPaint()设置图区的背景颜色。例如：

```
Plot plot = jfreechart.getPlot();
plot.setBackgroundPaint(Color.white);
```

我们可使用Paint接口的任何实现作为背景颜色的设置参数，其中有Color、GradientPaint（渐变颜色）和TexturePaint等。同时，我们也可以设置背景颜色为null。

9.3.4 设置背景图片

我们可以使用方法setBackgroundImage()为图区设置备有图片。

```
Plot plot = jfreechart.getPlot();
plot.setBackgroundImage(JFreeChart.INFO.getLogo());
```

默认的，图片充满图表的整个背景，图片失真。但我们可以改变图片不充满整个背景，使用方法是setBackgroundImageAlignment()。

```
plot.setBackgroundImageAlignment(Align.BOTTOM_RIGHT);
```

使用setBackgroundImageAlpha()方法，我们可以控制图片的透明度。如果我们希望图片充满这个图表区域，那么我们需要将背景图片添加到JFreeChart对象上（前面已经介绍过）。

9.4 轴属性

9.4.1 概述

使用JFreeChart创建的大部分图表都带有两个轴。X轴和Y轴。当然对于一些图表（比如饼图）根本就没有轴。对于使用轴的图表来说，图区使用Axis对象来管理轴。

9.4.2 获得轴对象引用

在你修改轴的属性之前，我们需要先获得一个轴的引用。图区类CategoryPlot和XYPlot类有两个方法getDomainAxis()和getRangeAxis()分别是获得X轴Y轴对象。这两个方法返回了一个ValueAxis对象的引用，除了在使用CategoryPlot的情况下，X轴使用的是CategoryAxis。代码如下：

```
// get an axis reference...
CategoryPlot plot = jfreechart.getCategoryPlot();
CategoryAxis domainAxis = plot.getDomainAxis();
// change axis properties...
domainAxis.setLabel("Categories");
domainAxis.setLabelFont(someFont);
```

CategoryAxis和ValueAxis类有许多不同的子类。有时我们需要将轴对象引用转化成具体的子类，为了获取更多具体的属性。如，如果我们想获得y轴为一个对象NumberAxis。代码如下：

```
XYPlot plot = jfreechart.getXYPlot();
NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
rangeAxis.setAutoRange(false);
```

9.4.3 设置轴标签

我们使用方法setLabel()可以改变轴的标签。如果我们不想在图表的轴上有标签，那么我们就设置为null即可。

我们可以使用Axis类定义的方法setLabelFont(), setLabelPaint(), 和setLabelInsets()改变标签的字体、颜色等内容。

9.4.4 改变周边标签显示方向

当图区在左侧或右侧画一个轴（水平轴）时，轴标签会自动旋转90度，以满足小空间的需要。如果我们希望标签也水平，我们需要修改标签的角度：

```
XYPlot plot = jfreechart.getXYPlot();
ValueAxis axis = plot.getRangeAxis();
axis.setLabelAngle(Math.PI / 2.0);
```

注意角度的表示使用弧度（PI为180度）。

9.4.5 隐藏刻度标签

隐藏某个轴的刻度标签：

```
CategoryPlot plot = jfreechart.getCategoryPlot();
ValueAxis axis = plot.getRangeAxis();
axis.setTickLabelsVisible(false);
```

对于CategoryAxis，方法setTickLabelsVisible(false)隐藏种类标签。

9.4.6 隐藏刻度符号

隐藏某个轴的刻度符号：

```
XYPlot plot = jfreechart.getXYPlot();
Axis axis = plot.getDomainAxis();
axis.setTickMarksVisible(false);
```

注意category轴没有刻度符号。

9.4.7 设置刻度尺寸

默认的，数值和日期会自动选择一个刻度尺寸，以便刻度标签不会重复显示。但我们也可以使用setTickUnit()方法设置我们自己的饿刻度单位。

9.4.8 指定标准的数值刻度单位

在NumberAxis类中，方法允许我们设置我们自己的刻度单位替代系统自动选择刻度danwi的机制。最普通的应用就是我们有一个仅仅显示整数的数轴。在实例中，我们不想让0.5或者0.25作为刻度单位。在NumberAxis类中有一个静态方法返回一系列的标准整数刻度单位：

```
XYPlot plot = jfreechart.getXYPlot();
NumberAxis axis = (NumberAxis) plot.getRangeAxis();
TickUnitSource units = NumberAxis.createIntegerTickUnits();
axis.setStandardTickUnits(units);
```

如果我们想控制标准的刻度单位时，我们可以自由定制自己的TickUnits集合。

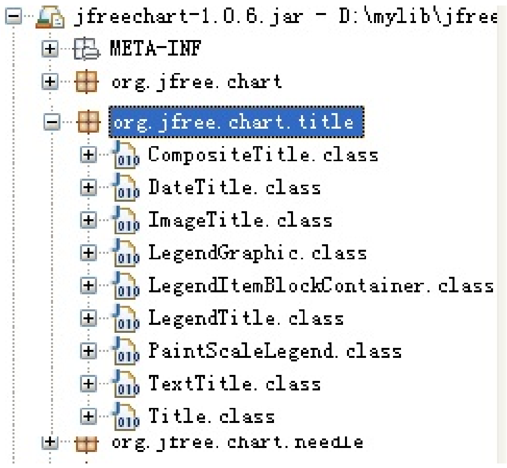
9.4.9 指定标准的日期刻度单位

类似于上一节内容，DateAxis类也有一个setStandardTickUnits()方法，来设置我们的刻度单位。方法createStandardDateTickUnits()为DateAxis返回了一个缺省的集合。同时我们也可以创建我们自己的标准日期刻度单位。

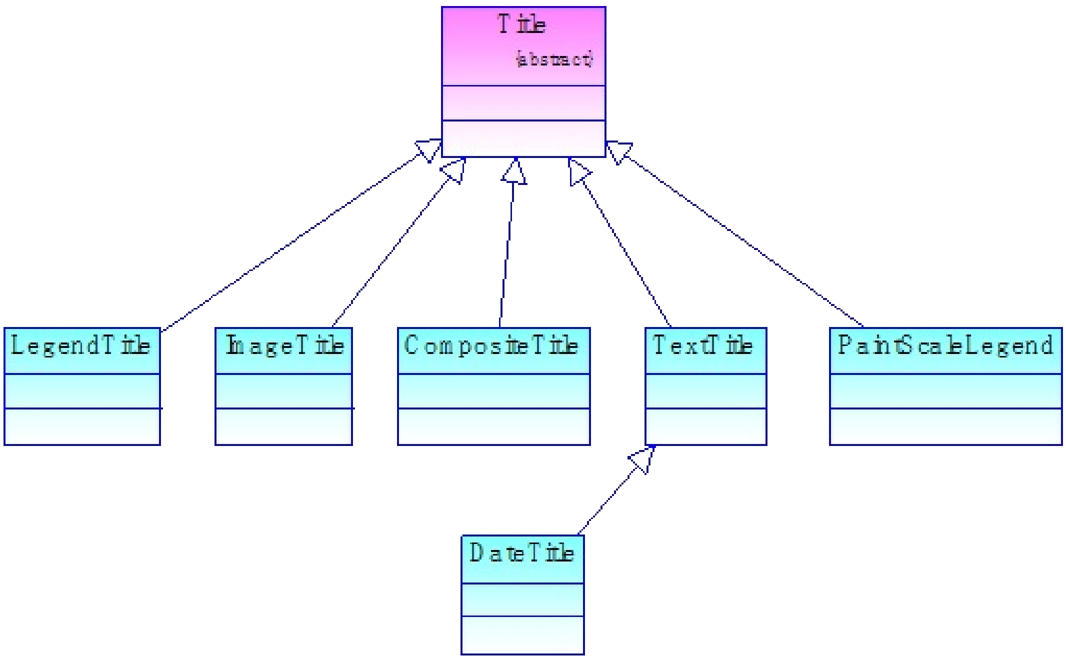
9.5 心得体会

9.5.1 Title子类如下图：

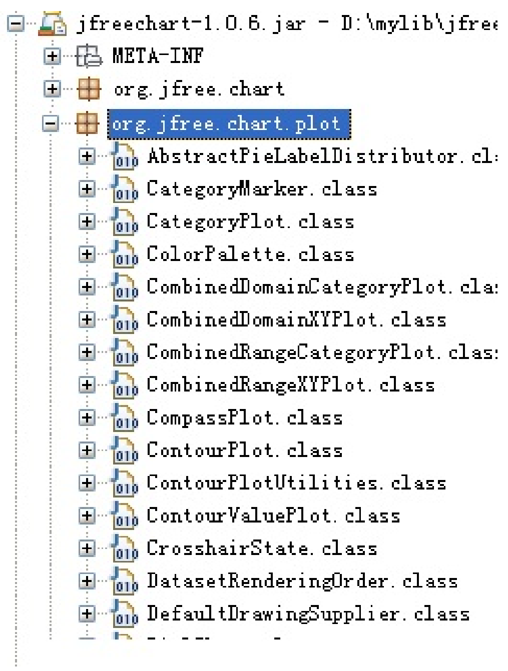
包图如下所示：



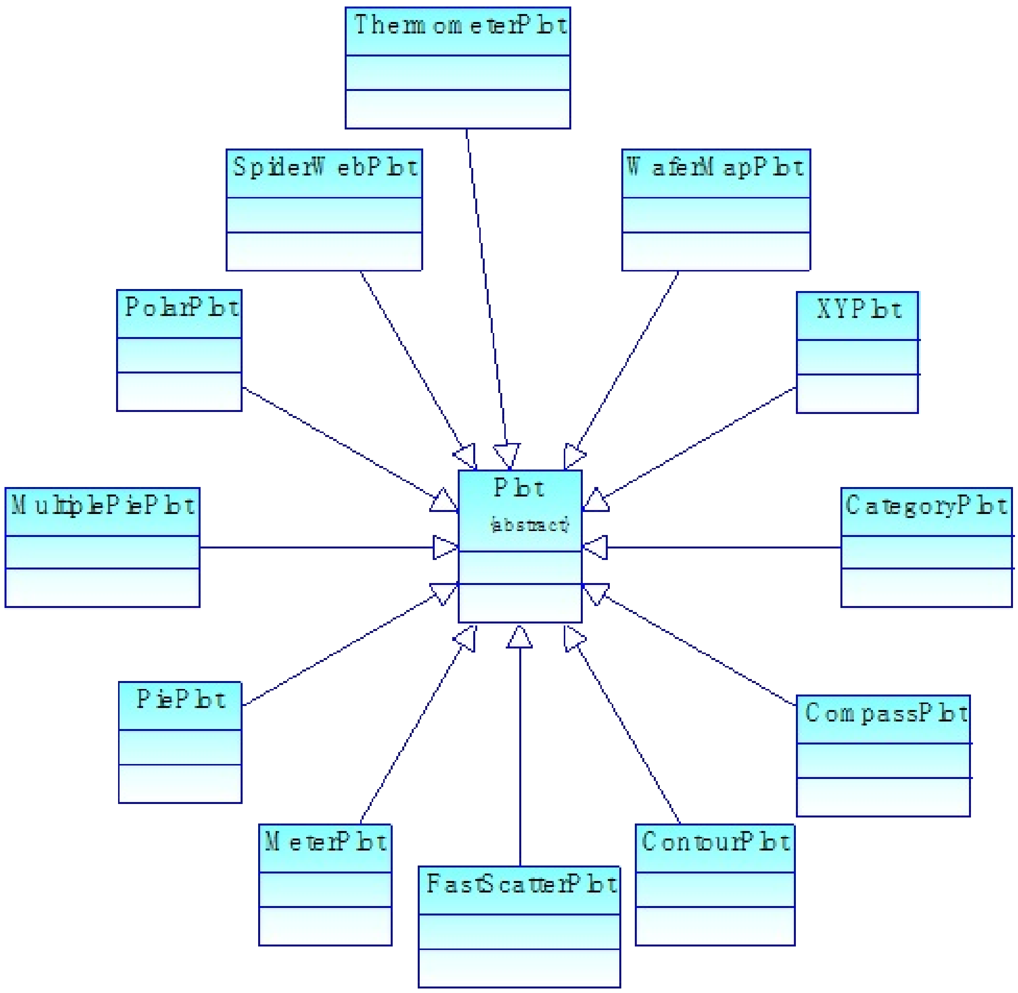
关系类图如下：



9.5.2 Plot类


jfreechart-1.0.6.jar - D:\mylib\jfre
META-INF
org.jfree.chart
org.jfree.chart.plot
AbstractPieLabelDistributor.cl
CategoryMarker.class
CategoryPlot.class
ColorPalette.class
CombinedDomainCategoryPlot.cla
CombinedDomainXYPlot.class
CombinedRangeCategoryPlot.clas
CombinedRangeXYPlot.class
CompassPlot.class
ContourPlot.class
ContourPlotUtilities.class
ContourValuePlot.class
CrosshairState.class
DatasetRenderingOrder.class
DefaultDrawingSupplier.class

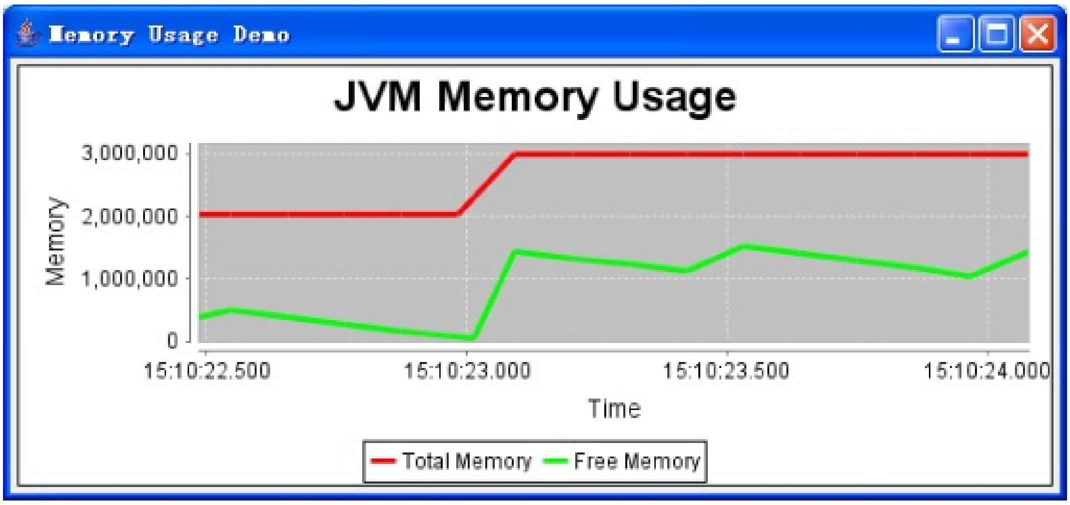
类结构图如下：



10 动态图（Dynamic Charts）

10.1 简介

为了说明使用JFreeChart创建“动态”的图表，本章节阐述了一个简单的应用说明这个过程，该应用为动态刷新JVM内存显示已使用和未使用情况。如下图10.1



如图10.1 一个简单的“动态”实例（参考：MemoryUsageDemo.java）。

10.2 知识背景

10.2.1 事件监听

JFreeChart使用监听机制来响应其他chart组件改变的响应。例如，不论数据源在何时发生更新，一个DatasetChangeEvent事件总被发送给已注册进数据源的监听器。

响应触发发生一系列事件：

- 图区监听到数据源改变的通知。如果需要更新轴的值，然后将PlotChangeEvent事件通知所有注册的监听器。
- 图表监听到图区更改事件的通知，然后将ChartChangeEvent事件通知给所有注册的监听器。
- 最后，ChartPanel接受到该面板上显示的图表的更改事件，ChartPanel根据响应的事件画出响应的图表——完全重新画，而不是仅仅更新数据。

所有图表或者其他子控件改变发生的事件过程都遵循上面的过程。

10.2.2 性能优化

关于性能优化，我们必须明白JFreeChart不会产生实时图表。每次数据源的更新，ChartPanel都需要重新画全部的图表。

性能优化通常是非常困难的。 比如，JFreeChart调用图像2D的API提取最新变更的点，从而只画更新的点。我们使用JFreeChart完成这个实时过程的实例将限制了“每秒产生页面”的数量。产生数量的大小是否是一个瓶颈的关键问题，主要取决我们我们画图表所依赖的数据，应用的环境和操作环境。

10.3 实例应用

10.3.1 概述

实例MemoryUsageDemo.java文档可以从下面的链接中获得：

http://www.object-refinery.com/jfreechart/premium/index.html

页面需要输入购买JFreeChart开发指南时需要的用户名和密码。

10.3.2 创建一个dataset

创建的实例代码数据源，包含了一个单一的时序集合，集合内有两个TimeSeries对象（一个是计算总内存，另一个是计算剩余内存）。代码如下：

```
this.total = new TimeSeries("Total", Millisecond.class);
this.total.setMaximumItemAge(30000);
this.free = new TimeSeries("Free", Millisecond.class);
this.free.setMaximumItemAge(30000);
TimeSeriesCollection dataset = new TimeSeriesCollection();
dataset.addSeries(this.total);
dataset.addSeries(this.free);
```

每个时间系列的maximumItemAge属性设置为30000毫秒（30秒）。因此任何时候添加到新系列的新数据，都是30秒前记录的老数据。

10.3.3 创建一个图表

图表的创建（定制）都遵循所有图表创建的标准模式。创建动态图也是一样，没有任何特殊的步骤。除了我们将autoRange属性设置为true之外。同时，这也有利于维护图表使用的数据源的引用。

10.3.4 更新一个dataset

在本实例演示中，通过向两个时序图添加的数据来更新数据源，该数据的添加有一个独立的线程Timer管理。代码如下：

```
class DataGenerator extends Timer implements ActionListener {
    DataGenerator(int i) {
        super(i, null);
        addActionListener(this);
    }
    public void actionPerformed(ActionEvent actionevent) {
        long l = Runtime.getRuntime().freeMemory();
        long l_0_ = Runtime.getRuntime().totalMemory();
        MemoryUsageDemo.this.addTotalObservation((double) l_0_);
        MemoryUsageDemo.this.addFreeObservation((double) l);
    }
}
```

注意JFreeChart在画图表和数据源更新代码之间没有使用线程同步，因此是不安全的。另一点需要注意的是，曾做过一个关于JFreeChart内存泄露问题的测试，将JFreeChart的实例在一个机器上连续运行6天。随着图表的不断更新，我们就可以看到垃圾收集器所产生的影响。六天之后，发现总内存的使用量保持不变。当JFreChart产生并丢弃的临时对象（垃圾对象）时，剩余可用内存减少了。增加的使用内存量是垃圾收集器工作时所使用的。

10.3.5 全部代码

下面是全部的实例代码：

```
/* MemoryUsageDemo - Decompiled by JODE
* Visit http://jode.sourceforge.net/
*/
package demo;
import java.awt.BasicStroke;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.BorderFactory;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.DateAxis;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.time.Millisecond;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
```

```
import org.jfree.ui.RectangleInsets;
public class MemoryUsageDemo extends JPanel {
    private static final long serialVersionUID = 67767128383594986
    private TimeSeries total;
    private TimeSeries free;
    /* synthetic */
    static Class class$org$jfree$data$time$Millisecond;
    class DataGenerator extends Timer implements ActionListener {
        private static final long serialVersionUID = 1L;
        DataGenerator(int i) {
            super(i, null);
            addActionListener(this);
        }
        public void actionPerformed(ActionEvent actionevent) {
            long l = Runtime.getRuntime().freeMemory();
            long l_0_ = Runtime.getRuntime().totalMemory();
            MemoryUsageDemo.this.addTotalObservation((double) l_0_
            MemoryUsageDemo.this.addFreeObservation((double) l);
        }
    }
    public MemoryUsageDemo(int i) {
        super(new BorderLayout());
        total = new TimeSeries(
            "Total Memory",
            (class$org$jfree$data$time$Millisecond == null ? (class
            : class$org$jfree$data$time$Millisecond));
        total.setMaximumItemAge((long) i);
        free = new TimeSeries(
            "Free Memory",
            (class$org$jfree$data$time$Millisecond == null ? (class
            : class$org$jfree$data$time$Millisecond));
        free.setMaximumItemAge((long) i);
        TimeSeriesCollection timeseriescollection = new TimeSeries(
        timeseriescollection.addSeries(total);
        timeseriescollection.addSeries(free);
        DateAxis dateaxis = new DateAxis("Time");
        NumberAxis numberaxis = new NumberAxis("Memory");
        dateaxis.setTickLabelFont(new Font("SansSerif", 0, 12));
        numberaxis.setTickLabelFont(new Font("SansSerif", 0, 12));
        dateaxis.setLabelFont(new Font("SansSerif", 0, 14));
        numberaxis.setLabelFont(new Font("SansSerif", 0, 14));
        XYLineAndShapeRenderer xylineandshaperenderer = new XYLineA
            true, false);
        xylineandshaperenderer.setSeriesPaint(0, Color.red);
        xylineandshaperenderer.setSeriesPaint(1, Color.green);
        xylineandshaperenderer.setSeriesStroke(0, new BasicStroke(3
        xylineandshaperenderer.setSeriesStroke(1, new BasicStroke(3
        XYPlot xyplot = new XYPlot(timeseriescollection, dateaxis,
            xylineandshaperenderer);
        xyplot.setBackgroundPaint(Color.lightGray);
        xyplot.setDomainGridlinePaint(Color.white);
        xyplot.setRangeGridlinePaint(Color.white);
        xyplot.setAxisOffset(new RectangleInsets(5.0, 5.0, 5.0, 5.0
```

```
        dateaxis.setAutoRange(true);
        dateaxis.setLowerMargin(0.0);
        dateaxis.setUpperMargin(0.0);
        dateaxis.setTickLabelsVisible(true);
        numberaxis.setStandardTickUnits(NumberAxis.createIntegerTic
        JFreeChart jfreechart = new JFreeChart("JVM Memory Usage",
            "SansSerif", 1, 24), xyplot, true);
        jfreechart.setBackgroundPaint(Color.white);
        ChartPanel chartpanel = new ChartPanel(jfreechart, true);
        chartpanel.setBorder(BorderFactory.createCompoundBorder(Bor
            .createEmptyBorder(4, 4, 4, 4), BorderFactory
            .createLineBorder(Color.black)));
        add(chartpanel);
    }
    private void addTotalObservation(double d) {
        total.add(new Millisecond(), d);
    }
    private void addFreeObservation(double d) {
        free.add(new Millisecond(), d);
    }
    public static void main(String[] strings) {
        JFrame jframe = new JFrame("Memory Usage Demo");
        MemoryUsageDemo memoryusagedemo = new MemoryUsageDemo(30000
        jframe.getContentPane().add(memoryusagedemo, "Center");
        jframe.setBounds(200, 120, 600, 280);
        jframe.setVisible(true);
        memoryusagedemo.new DataGenerator(100).start();
        jframe.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowevent) {
                System.exit(0);
            }
        });
    }
    /* synthetic */static Class class$(String string) {
        Class var_class;
        try {
            var_class = Class.forName(string);
        } catch (ClassNotFoundException classnotfoundexception) {
            throw new NoClassDefFoundError(classnotfoundexception.g
        }
        return var_class;
    }
}
```

11 图表工具条（Tooltips）

11.1 概述

JFreeChart为图表的每个组件提供了一套产生、收集和显示工具条的机制。本章主要介绍：

- 如何产生图表工具条（包括定制图表工具条）
- 如何收集图表工具条
- 如何显示图表工具条
- 如何隐藏图表工具条

11.2 创建图表工具条

如果我们需要使用图表工具条，我们首先确保所画的图表中已经产生图表工具条。我们可以为我们的图区或图区条目设置图表工具条产生器。在下面的相关章节里面，我们将了解如何为一个图表设置一个图表工具条。

11.2.1 饼图

饼图类PiePlot使用PieToolTipGenerator接口产生接口图表工具条。系统通过了该接口的一个标准实现类StandardPieToolTipGenerator。PiePlot设置图表工具条的方法如下：

```
public void setToolTipGenerator(PieToolTipGenerator generator);
```

该方法可以为饼图设置工具条产生器，如果设置null，则表示没有工具条。

11.2.2 generated.种类图

种类图表—包括JFreeChart创建最多的直方条形图—基于CategoryPlot类并使用CategoryItemRenderer来画每一个数据条目。Renderer使用接口CategoryToolTipGenerator的指定方法来获得图表工具条。为种类图区条目设置图表工具条产生器，使用类AbstractCategoryItemRenderer的方法：

```
public void setToolTipGenerator(CategoryToolTipGenerator generator)
```

该方法可以为饼图设置工具条产生器，如果设置null，则表示没有工具条。

11.2.3 XY图

XY图表—包括JFreeChart创建的散点图和时序图—基于类XYPlot并使用XYItemRenderer画出每一个数据条目。Renderer使用一个XYToolTipGenerator产生图表工具条。

设置XY图区条目的工具条，使用在AbstractXYItemRenderer定义的方法：

```
public void setToolTipGenerator(XYToolTipGenerator generator);
```

如果设置产生器为null，表示没有图表工具条产生器。

11.3 收集图表工具条

使用ChartRenderingInfo类可收集图表工具条信息，以及图表的其他信息。我们首先要向JFreeChart的draw()方法传入该类实例，否则图表工具条信息将不被记录（即便是产生器已经被注册到图区的数据条目中）。

幸运的是，ChartPanel会自动的处理图表工具条的收集。因此如果我们使用ChartPanel显示我们的图表，就不用担心图表工具条的收集—因为ChartPanel已经为我们收集了。

11.4 显示图表工具条

使用ChartPanel类创建我们的图表时，图表工具条会自动显示出来。并且你可以为图区（或者图区的renderer）设置一个图表工具条。

我们可是使用类的方法设置显示或隐藏图表工具条。方法如下：

```
public void setDisplayToolTips(boolean flag);
```

11.5 隐藏图表工具条

最有效的方式就是将图表工具条设置为null。确保没有任何图表工具条信息产生，这样可以节省内存同时提供处理速度（特别是对于大数据源时，非常有好处）。

我们可是使用上节讲的方法使用ChartPanele类设置图表工具条的隐藏。

11.6 定制图表工具条

我们可以通过相应的图表工具条产生器接口对每一个图表工具条进行文本的各种操作。

12 图表条目标签（Item Label）

12.1 简介

12.1.1 概述

对于大多数的图表类型来说，JFreeChart允许我们在图表的每个条目上、或者内部、或者附近显示条目标签。例如，下图12.1 在每个条形图上显示出了真实的值。

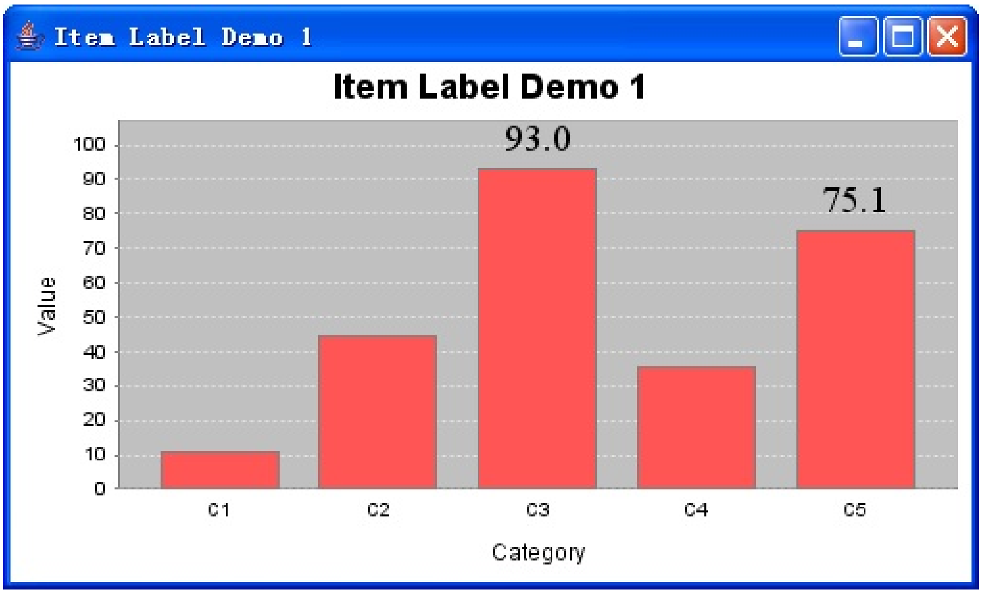


图12.1 显示数组的条形图（参考：）

本章主要讲述：

- 如何让条目标签可视（仅限于支持条目标签的图表类型）
- 如何改变条目标签的外观（字体和颜色）
- 如何指定条目标签的位置
- 如何定制条目标签的文本

忠告：我们使用上面的特征时，要谨慎。图表是期望用来分析总结数据的——如果我们觉得在图表上显示真实数据是非常有必要的话，那我们的数据应使用一个表格格式显示更为合适。

12.1.2 局限性

在当前版本JFreeChart中，条目标签的使用是有很多局限性的：

- 一些renderer不支持条目标签
- 轴范围的自动调节，忽略了条目标签的自动调整——如果图表的周围没有足够的空间（使用方法setUpperMargin()或setLowerMargin()进行了相应的调整），那么一些图表条目标签在图表上显示不出来。

相信，在以后的JFreeChart版本中，这些限制问题将被解决。

12.2 显示条目标签

12.2.1 概述

条目标签默认是不显示的，因此我们需要使用renderer进行创建和显示条目标签。这主要有以下两个步骤：

- 分配一个CategoryItemLabelGenerator或XYItemLabelGenerator给renderer—这是一个负责创建标签的对象。
- 在renderer里面设置一个标签可视的标志。可以针对全部系列进行设置，也可以针对具体的每一个系列进行设置。

此外，我们可以定制条目标签的位置、字体和颜色。在下面的章节里我们将详细的介绍。

12.2.2 创建一个条目标签并赋值

使用renderer分配的一个标签产生器创建条目标签（这与图表工具条的机制是相同的）。

下面代码说了将一个标签产生器指派给CategoryItemRenderer：

```
CategoryItemRenderer renderer = categoryplot.getRenderer();
CategoryItemLabelGenerator generator = new StandardCategoryItemLabe
renderer.setBaseItemLabelGenerator(generator);
```

同样的，将一个产生器指派给XYItemRenderer，代码如下：

```
XYPlot plot = (XYPlot) jfreechart.getPlot();
XYItemRenderer renderer = plot.getRenderer();
XYItemLabelGenerator generator = new StandardXYItemLabelGenerator(
    "{2}", new DecimalFormat("0.00"), new DecimalFormat("0.00"));
renderer.setBaseItemLabelGenerator(generator);
```

我们可以在标准产生器的构造函数中定制不同的行为。当然了，我们也可以创建我们总计的产生器，详见12.5.2章节。

12.2.3 所有的系列显示条目标签

方法renderer.setBaseItemLabelsVisible(false)是控制着条目标签的显示。对于CategoryItemRenderer：

```
CategoryItemRenderer renderer = categoryplot.getRenderer();
renderer.setBaseItemLabelsVisible(true);
```

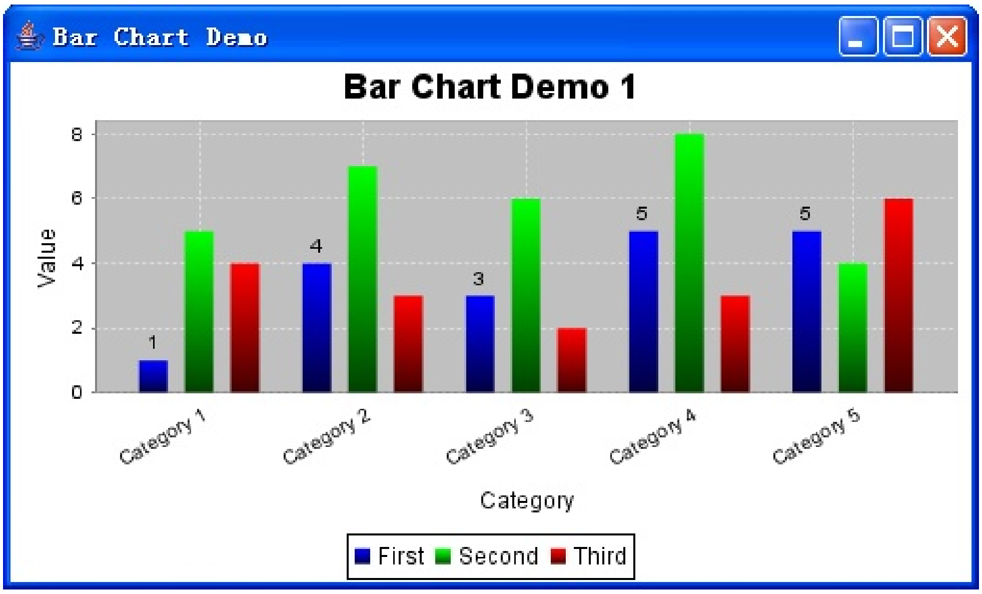
同样对于：XYItemRenderer

```
XYItemRenderer renderer = categoryplot.getRenderer();
renderer.setBaseItemLabelsVisible(true);
```

一旦设置，这个标志优先管理我们在所有地方对每一系列做的设置，主要为了应用每一系列的设置。我们可以设置个标志为null（见12.2.4章节）

12.2.4 为选择的系列显示条目标签

我们可以控制图表的每一个系列的条目标签是否显示。例如：如下图12.2仅显示第一系列条目标签。



如图12.2显示第一系列条目标签

下面代码可以设置如上效果：

```
CategoryItemRenderer renderer = categoryplot.getRenderer();
renderer.setBaseItemLabelGenerator(new StandardCategoryItemLabelGe
renderer.setBaseItemLabelsVisible(null); // clears the ALL series f
renderer.setSeriesItemLabelsVisible(0, true);
renderer.setSeriesItemLabelsVisible(1, false);
```

注意：上面代码中对全部的系列设置为null—这一点非常重要，因为全部系列的标志控制每一个系列的标志。

12.2.5 问题与解决

如果按照上面的步骤操作，你仍然未看见条目标签显示在图表上，那么我们从以下几个方面进行考虑：

- Rendererere必须需要一个标签产生器——这是一个用来创建每一个标签的文本条目的对象。
- 一些renderer不支持条目标签（具体参考renderer相关的文档）

12.3 条目标签外观

12.3.1 概述

我们可以通过改变条目的颜色、字体来改变图表条目标签的外观。正如其他renderer属性一样，属性的设置可以是全部的系列，可以是具体某一系列。

在JFreeChart目前的版本中，标签是月年个一个透明的背景画出来的。我们不能设置标签的背景颜色，也不能指定标签的边框。这些在以后的版本中会得到解决。

12.3.2 改变条目标签的字体

为了在所有的系列中改变条目标签的字体，我们可以使用下面的代码：

```
CategoryItemRenderer renderer = categoryplot.getRenderer();
renderer.setBaseItemLabelFont(new Font("黑体", Font.PLAIN, 20));
```

同样，也可以为单个系列设置字体：

```
// add settings for individual series...
renderer.setSeriesItemLabelFont(0, new Font("SansSerif", Font.PLAI
renderer.setSeriesItemLabelFont(1, new Font("SansSerif", Font.PLAI
```

注意：renderer.setBaseItemLabelFont(null)方法会出错。开发指南显示的代码有错误。

12.3.3 改变条目标签的颜色

改变条目标签的颜色，我们可以使用下面的代码：

```
CategoryItemRenderer renderer = categoryplot.getRenderer();
renderer.setBaseItemLabelPaint(Color.red);
```

同样的，可以为单独每一系列设置颜色：

```
// add settings for individual series...
renderer.setSeriesItemLabelPaint(0, Color.red);
renderer.setSeriesItemLabelPaint(1, Color.blue);
```

注意：renderer.setBaseItemLabelPaint(null);方法会出错。开发指南显示的代码有错误。

12.4 条目标签位置

12.4.1 概述

条目标签的位置是通过ItemLabelPosition对象的四个属性来控制的。

我们可以通过接口CategoryItemRenderer的方法来独立定义条目标签的正负点位置：

```
public void setBasePositiveItemLabelPosition(ItemLabelPosition posi
public void setBaseNegativeItemLabelPosition(ItemLabelPosition posi
```

理解这些属性如何影响独立标签的最终位置的关键是了解JFreeChart里面条目标签的特征。四个特征是：

- 条目标签点——决定标签的起始位置
- 文本点——标签里的文本相对于条目标签的位置。
- 旋转点——标签文本旋转的点位置
- 旋转角度——标签的旋转角度。

这些的详细描述在下一章详细介绍。

12.4.2 条目标签的位置

设置条目标签位置的目的，主要是为了找出标签在图表上贴向数据条目的一个点（x,y）位置。同时在画图表时，该标签也被画在该点处。更多的信息可以参考ItemLabelAnchor文档。

12.4.3 标签文本的位置

标签文本的位置，主要取决于上节讲的标签位置。我们可以讲标签文本在标签里设置在右上部、或左下部等，更多的信息参见TextAnchor文档。

运行JCommon包内的org.demo.package下面的DrawStringDemo应用，可以更好的理解标签文本在标签内是如何放置的。

12.4.4 标签旋转点

在标签上定义了一个旋转点，用于旋转标签。在DrawStringDemo实例中很好演示了这个特征。

12.4.5 标签旋转角度

旋转角度定义了标签沿旋转点旋转的角度。该角度为弧度。

12.5 定制条目标签文本

12.5.1 概述

定制条目标签文本，我们需要依赖用JFreeChart里的标签产生器来为条目标签创建文本。如果要想完全控制标签文本的控制，我们就需要编写自己的标签产生器，需要实现接口CategoryItemLabelGenerator。

在本章节里，我们对自定义标签器技术做了简要的讲述，然后用两个实例来说明该技术过程。

12.5.2 实现一个自定义的标签产生器

开发一个自定义标签产生器，我们需要写一个类，该类必须实现CategoryItemLabelGenerator接口里的方法。

```
public String generateLabel(CategoryDataset dataset, int series,int
```

该renderer调用该方法获得一个标签的字符串，并且将该字符串传入到当前条目的CategoryDataset、序列和种类。这就意味着创建这个标签时，我们拥有完全的访问权限。

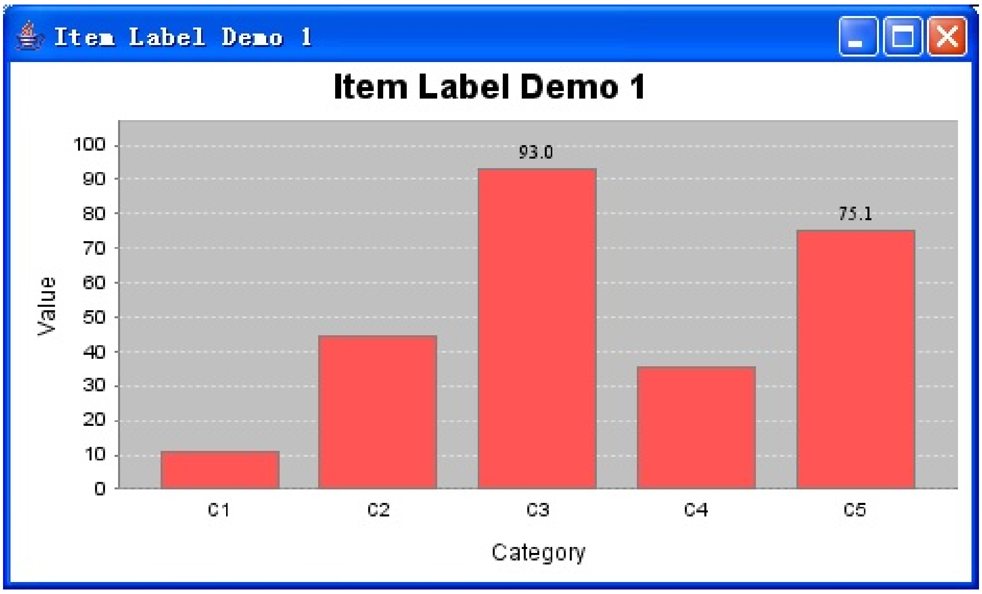
该方法可以返回任意字符串，因此我们格式化这个字符串。如果我们不想显示标签，可以设置为null。

在下面的两个例子中很好的说明了这个特征。

12.6 实例1

12.6.1 概述

在第一个实例中，目的就是当当条目的值大于某个限定的值时，就显示该标签。如图12.3所示。



如图12.3 超过某个限定值显示条目标签的实例

做到这一点并不困难，需要做以下工作：

- 写一个实现接口CategoryItemLabelGenerator的类，并且实现generateItemLabel()方法。该方法实现如果条目的值小于限定值时，返回null。
- 创建该类的实例，将该实例使用renderer的方法setLabelGenerator()设置到renderer中去。

12.6.2 源代码

```
package demo;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.labels.AbstractCategoryItemLabelGenerator;
```

```
import org.jfree.chart.labels.CategoryItemLabelGenerator;
import org.jfree.chart.labels.StandardCategoryItemLabelGenerator;
import org.jfree.chart.labels.StandardXYItemLabelGenerator;
import org.jfree.chart.labels.XYItemLabelGenerator;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.category.CategoryItemRenderer;
import org.jfree.chart.renderer.xy.XYItemRenderer;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;
public class ItemLabelDemo1 extends ApplicationFrame {
    static class LabelGenerator extends AbstractCategoryItemLabelGe
        implements CategoryItemLabelGenerator {
        private double threshold;
        public LabelGenerator(double d) {
            super("", NumberFormat.getInstance());
            threshold = d;
        }
        public String generateLabel(CategoryDataset categorydataset
            int i_0_) {
            String string = null;
            Number number = categorydataset.getValue(i, i_0_);
            if (number != null) {
                double d = number.doubleValue();
                if (d &gt; threshold)
                string = number.toString();
            }
            return string;
        }
    }
    public ItemLabelDemo1(String string) {
        super(string);
        CategoryDataset categorydataset = createDataset();
        JFreeChart jfreechart = createChart(categorydataset);
        ChartPanel chartpanel = new ChartPanel(jfreechart);
        chartpanel.setPreferredSize(new Dimension(500, 270));
        setContentPane(chartpanel);
    }
    private static CategoryDataset createDataset() {
        DefaultCategoryDataset defaultcategorydataset = new Default
        defaultcategorydataset.addValue(11.0, "S1", "C1");
        defaultcategorydataset.addValue(44.3, "S1", "C2");
        defaultcategorydataset.addValue(93.0, "S1", "C3");
        defaultcategorydataset.addValue(35.6, "S1", "C4");
        defaultcategorydataset.addValue(75.1, "S1", "C5");
        return defaultcategorydataset;
    }
    private static JFreeChart createChart(CategoryDataset category
        JFreeChart jfreechart = ChartFactory.createBarChart(
            "Item Label Demo 1", "Category", "Value", categorydatas
```

```
            PlotOrientation.VERTICAL, false, true, false);
            jfreechart.setBackgroundPaint(Color.white);
            CategoryPlot categoryplot = (CategoryPlot) jfreechart.getPl
            categoryplot.setBackgroundPaint(Color.lightGray);
            categoryplot.setDomainGridlinePaint(Color.white);
            categoryplot.setRangeGridlinePaint(Color.white);
            NumberAxis numberaxis = (NumberAxis) categoryplot.getRangeA
            numberaxis.setUpperMargin(0.15);
            return jfreechart;
        }
        public static JPanel createDemoPanel() {
            JFreeChart jfreechart = createChart(createDataset());
            return new ChartPanel(jfreechart);
        }
        public static void main(String[] strings) {
            ItemLabelDemo1 itemlabeldemo1 = new ItemLabelDemo1("Item La
            itemlabeldemo1.pack();
            RefineryUtilities.centerFrameOnScreen(itemlabeldemo1);
            itemlabeldemo1.setVisible(true);
        }
    }
```

12.7 实例2

12.7.1 概述

在本实例中，目的是在每个系列的标签上显示出值和百分比值（这个百分比值，这个系列在某一部分的条形直方图或全部条形直方图的总值中的比值）。如下图12.4所示。

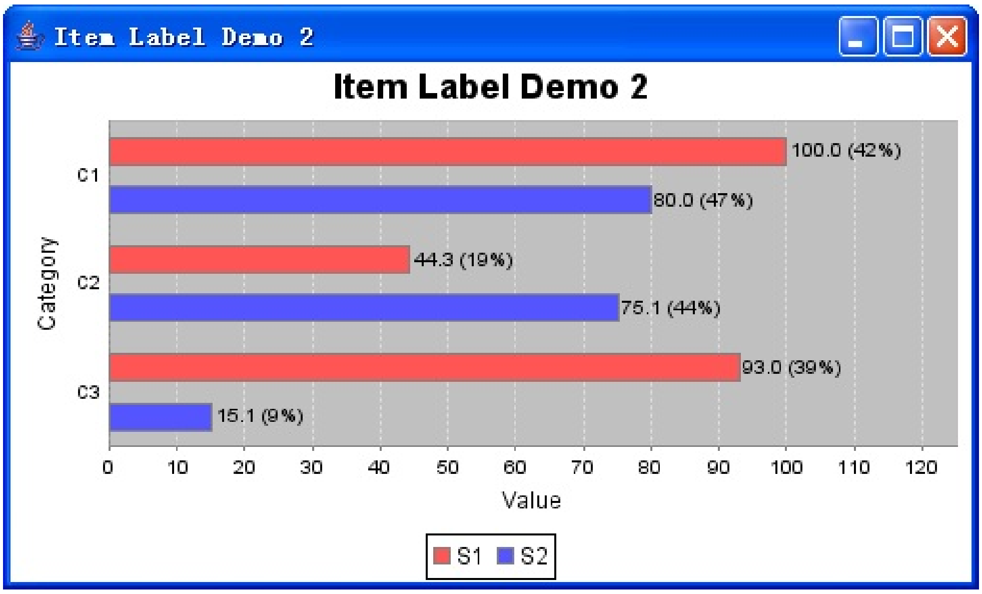


图12.4 带有比值的直方图

该实现中，标签产生器计算出百分比。如果传入构造函数的是一个种类索引，那么这个百分比的基数就是指定种类的当前系列的值。如果种类索引是无效的，那么这个基数就是指定种类的全部系列总和。

标签产生器会默认创建一个百分比格式——一种比较成熟的格式，提供格式化能力。

12.7.2 源代码

```
package demo;
import java.awt.Color;
import java.awt.Dimension;
import java.text.NumberFormat;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.AxisLocation;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.labels.AbstractCategoryItemLabelGenerator;
import org.jfree.chart.labels.CategoryItemLabelGenerator;
```

```
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.category.CategoryItemRenderer;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;
public class ItemLabelDemo2 extends ApplicationFrame {
    static class LabelGenerator extends AbstractCategoryItemLabelGe
    implements CategoryItemLabelGenerator {
        private Integer category;
        private NumberFormat formatter = NumberFormat.getPercentIns
        public LabelGenerator(int i) {
            this(new Integer(i));
        }
        public LabelGenerator(Integer integer) {
            super("", NumberFormat.getInstance());
            category = integer;
        }
        public String generateLabel(CategoryDataset categorydataset
            int i_0_) {
            String string = null;
            double d = 0.0;
            if (category != null) {
                Number number = categorydataset
                .getValue(i, category.intValue());
                d = number.doubleValue();
            } else
                d = calculateSeriesTotal(categorydataset, i);
            Number number = categorydataset.getValue(i, i_0_);
            if (number != null) {
                double d_1_ = number.doubleValue();
                string = (number.toString() + " (" + formatter.forn
            }
            return string;
        }
        private double calculateSeriesTotal(CategoryDataset categor
        int i) {
        double d = 0.0;
        for (int i_2_ = 0; i_2_ &lt; categorydataset.getColumnCount
        Number number = categorydataset.getValue(i, i_2_);
        if (number != null)
        d += number.doubleValue();
        }
        return d;
        }
    }
    public ItemLabelDemo2(String string) {
        super(string);
        CategoryDataset categorydataset = createDataset();
        JFreeChart jfreechart = createChart(categorydataset);
        ChartPanel chartpanel = new ChartPanel(jfreechart);
        chartpanel.setPreferredSize(new Dimension(500, 270));
```

```
            setContentPane(chartpanel);
        }
        private static CategoryDataset createDataset() {
            DefaultCategoryDataset defaultcategorydataset = new Default
            defaultcategorydataset.addValue(100.0, "S1", "C1");
            defaultcategorydataset.addValue(44.3, "S1", "C2");
            defaultcategorydataset.addValue(93.0, "S1", "C3");
            defaultcategorydataset.addValue(80.0, "S2", "C1");
            defaultcategorydataset.addValue(75.1, "S2", "C2");
            defaultcategorydataset.addValue(15.1, "S2", "C3");
            return defaultcategorydataset;
        }
        private static JFreeChart createChart(CategoryDataset category
            JFreeChart jfreechart = ChartFactory.createBarChart(
                "Item Label Demo 2", "Category", "Value", categorydatas
            PlotOrientation.HORIZONTAL, true, true, false);
            jfreechart.setBackgroundPaint(Color.white);
            CategoryPlot categoryplot = (CategoryPlot) jfreechart.getPl
            categoryplot.setBackgroundPaint(Color.lightGray);
            categoryplot.setDomainGridlinePaint(Color.white);
            categoryplot.setRangeGridlinePaint(Color.white);
            categoryplot.setRangeAxisLocation(AxisLocation.BOTTOM_OR_LE
            NumberAxis numberaxis = (NumberAxis) categoryplot.getRange
            numberaxis.setUpperMargin(0.25);
            CategoryItemRenderer categoryitemrenderer = categoryplot.ge
            categoryitemrenderer.setBaseItemLabelsVisible(true);
            categoryitemrenderer.setBaseItemLabelGenerator(new LabelGen
                (Integer) null));
            return jfreechart;
        }
        public static JPanel createDemoPanel() {
            JFreeChart jfreechart = createChart(createDataset());
            return new ChartPanel(jfreechart);
        }
        public static void main(String[] strings) {
            ItemLabelDemo2 itemlabeldemo2 = new ItemLabelDemo2("Item La
            itemlabeldemo2.pack();
            RefineryUtilities.centerFrameOnScreen(itemlabeldemo2);
            itemlabeldemo2.setVisible(true);
        }
    }
```

13 多轴和数据源图表（Multi Axis and Dataset）

13.1 简介

J FreeChart在CategoryPlot和XYPlot类中支持多轴和数据源显示。我们利用这个特征可以在一个图表上显示两个或多个数据源数据，但对于数据包含的数据有巨大差距时留有一定的余地。如图13.1所示。

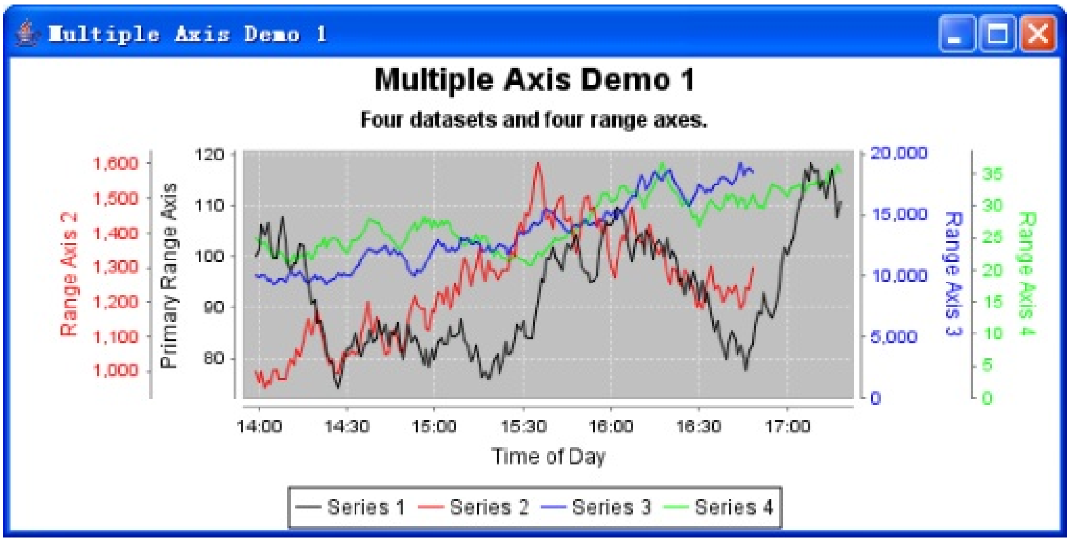


图13.1 具有多轴的图表

典型的，使用JFreeChart构建图表时，图表有一个单数据源、单renderer、单X/Y轴的图区最为常见。然而，在一个图区上添加多个数据源、多个renderer和多个轴也是可能的。在本章的实例中，展示了如何在一个图区上显示其他额外的数据源、renderer和轴。

13.2 实例

13.2.1 简介

MultipleAxisDemo1.java例子提供了一个很好的实例演示如何在一个图表上创建多轴的应用。本章在每一步的代码中提供了很多建议，详见后面的章节。

13.2.2 创建一个图表

创建一个具有多轴、多数据源、多renderer的图表，我们首先要创建一个常规的图表（例如使用ChartFactory类创建）。在本实例中，创建了一个时序图，代码如下：

```
XYDataset dataset1 = createDataset("Series 1", 100.0, new Minute(),
JFreeChart chart = ChartFactory.createTimeSeriesChart(
    "Multiple Axis Demo 1",
    "Time of Day",
    "Primary Range Axis",
    dataset1,
    true,
    true,
    false
);
```

13.2.3 添加额外的轴

如果在图区上添加额外的轴，我们使用setRangeAxis()方法来添加：

```
NumberAxis numberaxis = new NumberAxis("Range Axis 2"); xyplot.setR
xyplot.setRangeAxisLocation(1, AxisLocation.BOTTOM_OR_LEFT);
```

方法setRangeAxis()是用来添加图区的轴，注意轴的索引1已经被使用——我们添加其他轴时，通过增加该索引来添加新轴。方法setRangeAxisLocation()允许我们指定轴出现的位置（使用AxisLocation类）。我们添加的轴可以跟主坐标轴同一边，或者在对立边。例如：如果指定的是AxisLocation.BOTTOM_OR_LEFT，这意味着如果图区的方向是垂直的话，将在右边添加了一个Y轴，如果图区的方向是水平的话，将在底部添加一个Y轴。

在这里，图表上每一添加多余的数据源，因此如果我们显示该图表，我们将看到图上显示多轴，但轴上无数据显示。

13.2.4 添加一个额外的数据源

在图区上添加一个额外的数据源，使用setDataset()方法：

```
XYDataset xydataset_0_ = createDataset("Series 2", 1000.0,
new Minute(), 170);
xyplot.setDataset(1, xydataset_0_);
```

缺省的，数据源将使用主轴来显示数据。如果使数据源在另外的轴上显示数据，需使用方法mapDatasetToDomainAxis()和mapDatasetToRangeAxis()。这两个方法接受两个参数，第一个参数是数据源的索引，第二个是轴的索引。

13.2.5 添加一个额外的renderer

当我们添加一个数据源时，通常为该数据源添加一个附加的renderer也是非常有意义的。使用方法setRenderer()：

```
StandardXYItemRenderer standardxyitemrenderer = new StandardXYItemR
xyplot.setRenderer(1, standardxyitemrenderer);
```

方法的第一个参数为上节中添加的数据源的索引。注意：如果我们不想为数据源指定一个附加的rendere，系统将默认使用主renderer，这样系列的颜色就会在主数据源和附加数据源之间共享。

13.3 建议和技巧

当我们使用多轴图表时，我们需要为系列对应的轴提供一些可视化的建议。比如在例子MultipleAxisDemo1.java中轴标签的颜色与系列颜色是相匹配的。

可以从下面的实例中学习更多的技巧：

- DualAxisDemo1.java
- DualAxisDemo2.java
- DualAxisDemo3.java
- DualAxisDemo4.java
- MultipleAxisDemo1.java
- MultipleAxisDemo2.java
- MultipleAxisDemo3.java

14 组合图表（Combined Charts）

14.1 简介

JFreeChart支持几个图区类（可以管理着多个子类）组合而成的图表。图区类可以管理几个子类：

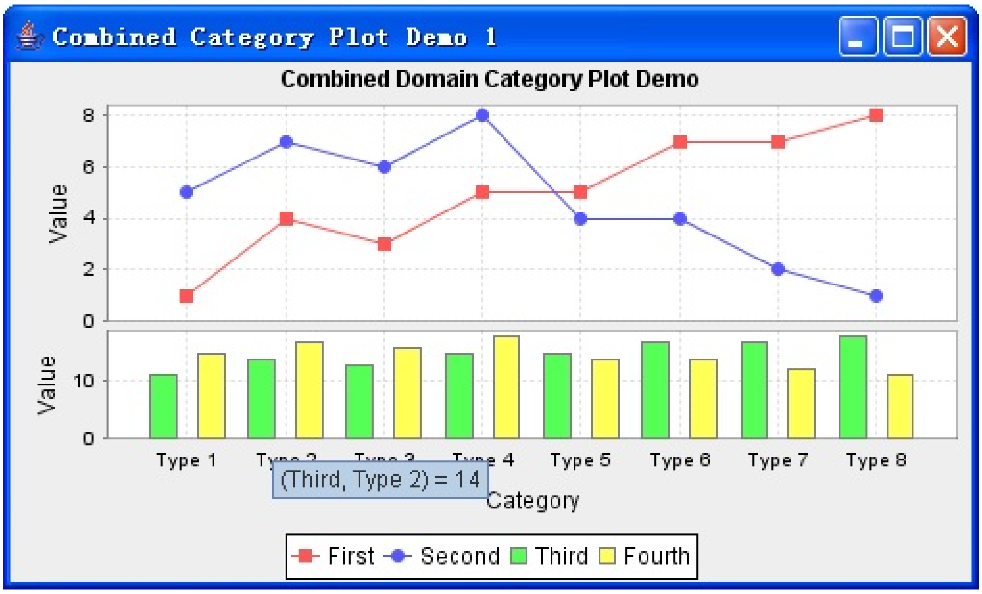
- `CombinedDomainCategoryPlot` / `CombinedRangeCategoryPlot`
- `CombinedDomainXYPlot` / `CombinedRangeXYPlot`；

本章使用几个实例说明了JFreeChart创建组合图表时的便利性。

14.2 组合X种类图区

14.2.1 概述

组合主域种类图区就是在一个图区上显示两个或者多个子图区（CategoryPlot实例），共享一个X轴的图区。每个子图区维护自己的Y轴。实例如图14.1所示。



如图14.1 组合X种类图区(共享X轴)

显示图表可以是水平的，也可以是垂直方向的——实例演示的是垂直的图表。

14.2.2 构建图表

提供了一个很好的例子，演示如何创建该图表的类型。关键的步骤是创建CombinedDomainCategoryPlot实例，然后添加两个子图区：

```
CategoryAxis domainAxis = new CategoryAxis("Category");
CombinedDomainCategoryPlot plot = new CombinedDomainCategoryPlot(do
plot.add(subplot1, 2);
plot.add(subplot2, 1);
JFreeChart result = new JFreeChart(
    "Combined Domain Category Plot Demo",
    new Font("SansSerif", Font.BOLD, 12),
    plot,
    true
);
```

注意，我们subplot1添加码值时是2（方法add()的第二个参数），而subplot1添加的是1呢？因为这控制着分配给各个图区的空间大小。

子图区的CategoryPlot实例对象将它们的X轴设置为null。例如在演示的实例中，代码如下：

```
CategoryDataset dataset1 = createDataset1();
NumberAxis rangeAxis1 = new NumberAxis("Value");
rangeAxis1.setStandardTickUnits(NumberAxis.createIntegerTickUnits()
LineAndShapeRenderer renderer1 = new LineAndShapeRenderer();
renderer1.setBaseToolTipGenerator(new StandardCategoryToolTipGenera
CategoryPlot subplot1 = new CategoryPlot(dataset1, null, rangeAxis1
subplot1.setDomainGridlinesVisible(true);
CategoryDataset dataset2 = createDataset2();
NumberAxis rangeAxis2 = new NumberAxis("Value");
rangeAxis2.setStandardTickUnits(NumberAxis.createIntegerTickUnits()
BarRenderer renderer2 = new BarRenderer();
renderer2.setBaseToolTipGenerator(new StandardCategoryToolTipGenera
CategoryPlot subplot2 = new CategoryPlot(dataset2, null, rangeAxis2
subplot2.setDomainGridlinesVisible(true);
```

14.3 组合Y种类图区

14.3.1 概述

一个组合Y种类图区就是一个图区显示两个或两个以上的子图区（CategoryPlot实例），共享Y轴。如果14.2.

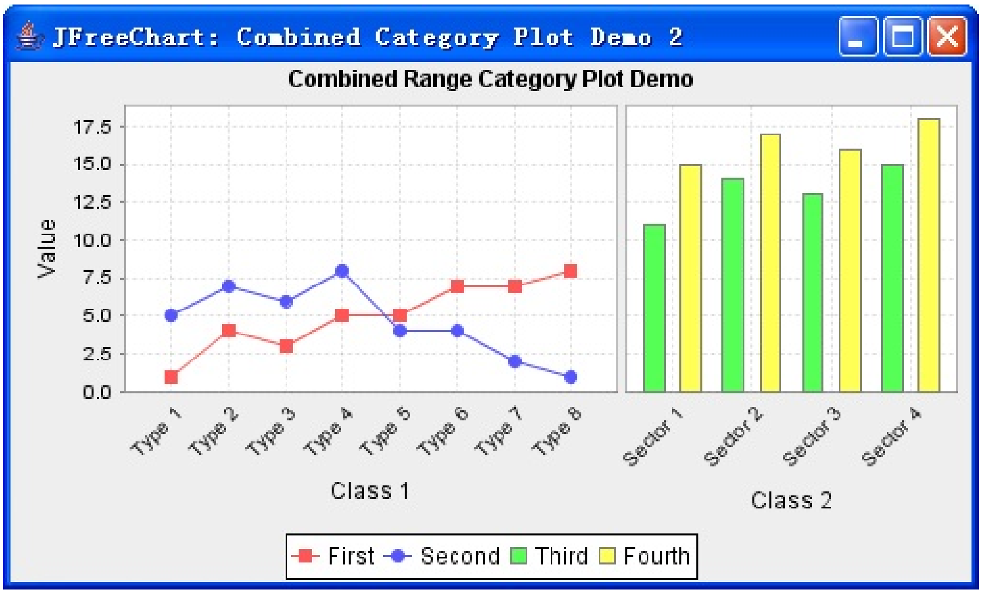


图14.2 组合Y种类图区

该图表可以水平显示也可以垂直显示（本例是垂直显示）。

14.3.2 构建图表

实例演示了如何创建该类型图表。关键的步骤是创建一个实例，然后添加两个子图区：

```
ValueAxis rangeAxis = new NumberAxis("Value");
```

CombinedRangeCategoryPlot plot = new CombinedRangeCategoryPlot(rangeAxis); plot.add(subplot1, 3); plot.add(subplot2, 2); JFreeChart result = new JFreeChart("Combined Range Category Plot Demo", new Font("SansSerif", Font.BOLD, 12), plot, true);

```
注意添加的子图区subplot1什么码值是3而子图区subplot2码值是2呢。这是因为该值控

子图区是CategoryPlot实例，将Y轴设置为null。例如，在本实例演示的代码如下：
```

```
CategoryDataset dataset1 = createDataset1(); CategoryAxis domainAxis1 = new
CategoryAxis("Class 1");
domainAxis1.setCategoryLabelPositions(CategoryLabelPositions.UP 45);
domainAxis1.setMaxCategoryLabelWidthRatio(5.0f); LineAndShapeRenderer
renderer1 = new LineAndShapeRenderer();
renderer1.setBaseToolTipGenerator(new StandardCategoryToolTipGenerator());
CategoryPlot subplot1 = new CategoryPlot(dataset1, domainAxis1, null,
renderer1); subplot1.setDomainGridlinesVisible(true); CategoryDataset dataset2 =
createDataset2(); CategoryAxis domainAxis2 = new CategoryAxis("Class 2");
domainAxis2.setCategoryLabelPositions(CategoryLabelPositions.UP 45);
domainAxis2.setMaxCategoryLabelWidthRatio(5.0f); BarRenderer renderer2 =
new BarRenderer(); renderer2.setBaseToolTipGenerator(new
StandardCategoryToolTipGenerator()); CategoryPlot subplot2 = new
CategoryPlot(dataset2, domainAxis2, null, renderer2);
subplot2.setDomainGridlinesVisible(true);
```

14.4 组合X-XY图区

14.4.1 概述

组合X-XY图区就是一个图区显示两个或者多个子图区（XYPlot实例），共享一个X轴。每一个子图区维护自己的Y轴。如下图14.3所示.

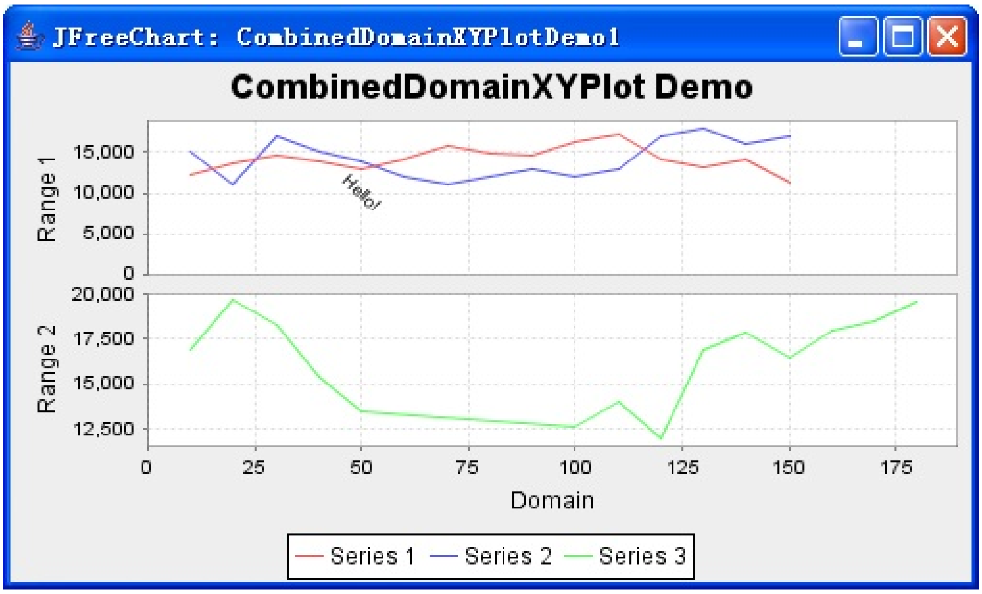


图14.3 组合X-XY图区（参见：CombinedXYPlotDemo5.java）

图区可能水平显示也可能垂直显示（本例子中垂直显示）。

14.4.2 构建图表

CombinedXYPlotDemo5.java实例演示了如何创建该类型的图表。关键的步骤是创建一个实例CombinedDomainXYPlot，并在该实例上添加两个子图区：

```
CombinedDomainXYPlot plot = new CombinedDomainXYPlot(new NumberAxis
plot.setGap(10.0);
plot.add(subplot1, 1);
plot.add(subplot2, 1);
plot.setOrientation(PlotOrientation.VERTICAL);
return new JFreeChart(
    "CombinedDomainXYPlot Demo",
    JFreeChart.DEFAULT_TITLE_FONT, plot, true
);
```

注意两个图区的码值为什么都是1呢？因为该数值控制着每个图区分配的空间大小。

子图区是XYPlot实例，将自己的X轴设置为null。例如，下面的代码演示了这个特征：

```
XYDataset data1 = createDataset1();
XYItemRenderer renderer1 = new StandardXYItemRenderer();
NumberAxis rangeAxis1 = new NumberAxis("Range 1");
XYPlot subplot1 = new XYPlot(data1, null, rangeAxis1, renderer1);
subplot1.setRangeAxisLocation(AxisLocation.BOTTOM OR LEFT);
XYTextAnnotation annotation = new XYTextAnnotation("Hello!", 50.0,
annotation.setFont(new Font("SansSerif", Font.PLAIN, 9));
annotation.setRotationAngle(Math.PI / 4.0);
subplot1.addAnnotation(annotation);
// create subplot 2...
XYDataset data2 = createDataset2();
XYItemRenderer renderer2 = new StandardXYItemRenderer();
NumberAxis rangeAxis2 = new NumberAxis("Range 2");
rangeAxis2.setAutoRangeIncludesZero(false);
XYPlot subplot2 = new XYPlot(data2, null, rangeAxis2, renderer2);
subplot2.setRangeAxisLocation(AxisLocation.TOP OR LEFT);
```

14.5 组合Y-XY图区

14.5.1 概述

组合Y-XY图区就是一个图区显示两个或者多个子图区（XYPlot实例），共享一个Y轴。每一个子图区维护自己的X轴。如下图14.4所示.

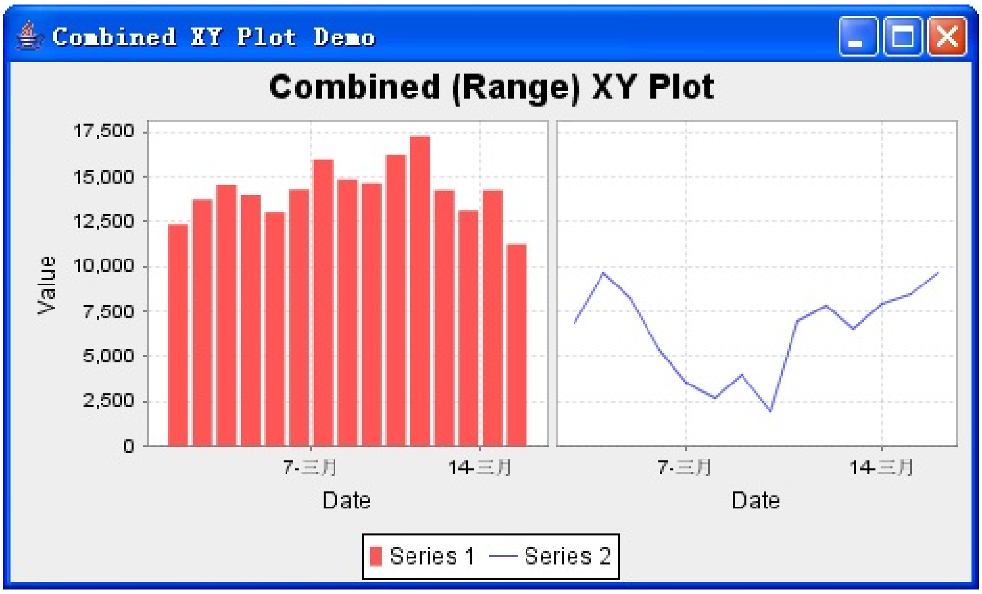


图14.4 组合Y-XY图区（参见：CombinedXYPlotDemo5.java）

图区可能水平显示也可能垂直显示（本例子中垂直显示）。

14.5.2 构建图表

CombinedXYPlotDemo2java实例演示了如何创建该类型的图表。关键的步骤是创建一个实例CombinedRangeXYPlot，并在该实例上添加两个子图区：

```
// create the plot...
CombinedRangeXYPlot plot = new CombinedRangeXYPlot(new NumberAxis('
plot.add(xyplot, 1);
plot.add(xyplot_0_, 1);
return new JFreeChart(
    "Combined (Range) XY Plot",
    JFreeChart.DEFAULT_TITLE_FONT, plot, true
);
```

注意两个图区的码值为什么都是1呢？因为该数值控制着每个图区分配的空间大小。

子图区是XYPlot实例，将自己的X轴设置为null。例如，下面的代码演示了这个特征：

```
IntervalXYDataset intervalxydataset = createDataset1();
XYBarRenderer xybarrenderer = new XYBarRenderer(0.2);
xybarrenderer.setBaseToolTipGenerator(new StandardXYToolTipGenerato
    "{0}: ({1}, {2})", new SimpleDateFormat("d-MMM-yyyy"),
    new DecimalFormat("0,000.0")));
XYPlot xyplot = new XYPlot(intervalxydataset, new DateAxis("Date"),
    null, xybarrenderer);
XYDataset xydataset = createDataset2();
StandardXYItemRenderer standardxyitemrenderer = new StandardXYItemR
standardxyitemrenderer
    .setBaseToolTipGenerator(new StandardXYToolTipGenerator(
    "{0}: ({1}, {2})", new SimpleDateFormat("d-MMM-yyyy"),
    new DecimalFormat("0,000.0")));
XYPlot xyplot_0_ = new XYPlot(xydataset, new DateAxis("Date"), null
    standardxyitemrenderer);
```

15 数据源和JDBC(Dataset And JDBC)

15.1 简介

本章节，主要讲述使用JDBC从数据库表中获得数据的几种数据源方法：

- JDBCPieDataset
- JDBCCategoryDataset
- JDBCXYDataset

15.2 关于JDBC

JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。

15.3 样本数据

我们再看一下实际运行中的JDBC数据源。我们需要在一个测试数据库中创建一些样本数据。

下面列出了创建饼图、直方条形图和时序图的样本数据。

创建饼图可以使用下面数据（在表中称谓饼数据）：

CATEGORY	VALUE
London	54.3
New York	43.4
Paris	17.9

同样，直方条形图使用下面数据创建（在表中称谓种类数据）：

CATEGORY	SERIES1	SERIES2	SERIES3
London	54.3	32.1	53.4
New York	43.4	54.3	75.2
Paris	17.9	34.8	37.1

最后，时序图表的使用的数据如下（在表中称谓xy数据）：

X	SERIES1	SERIES2	SERIES3
1-Aug-2002	54.3	32.1	53.4
2-Aug-2002	43.4	54.3	75.2
3-Aug-2002	39.6	55.9	37.1
4-Aug-2002	35.4	55.2	27.5
5-Aug-2002	33.9	49.8	22.3
6-Aug-2002	35.2	48.4	17.7
7-Aug-2002	38.9	49.7	15.3
8-Aug-2002	36.3	44.4	12.1
9-Aug-2002	31.0	46.3	11.0

我们可以创建一个测试数据库，包含上面的表。这里我们创建一个jfreechartdb数据库。

在下一章节里，我们使用PostgreSQL创建数据。如果使用的是不同的数据库系统，我们需要对这个过程做一些修改。

15.4 PostgreSQL

15.4.1 关于PostgreSQL

PostgreSQL是一个非常强大的面向关系的数据库服务系统，是一个开源的分布式系统。我们可以从下面链接获得更多的信息：

http://www.postgresql.org

注意尽管PostgreSQL是开源的，但它具有其他大型商业关系数据库系统的大部分特征。这里鼓励你安装，并使用它。

15.4.2 创建一个新的数据库

首先，登录数据库管理系统，创建一个名为jfreechartdb的数据库。

```
CREATE DATABASE jfreechartdb;
```

其次，创建一个jfreechart用户：

```
CREATE USER jfreechart WITH PASSWORD 'password';
```

JDBC可以使用这个用户名和密码进行数据库的连接。

15.4.3 创建饼图数据

创建饼图数据库表：

```
CREATE TABLE piedata1 (
    category VARCHAR(32),
    value FLOAT
);
```

加入样本数据：

```
INSERT INTO piedata1 VALUES ('London', 54.3);
INSERT INTO piedata1 VALUES ('New York', 43.4);
INSERT INTO piedata1 VALUES ('Paris', 17.9);
```

15.4.4 创建种类图表数据

创建种类图数据库表：

```
CREATE TABLE categorydata1 (
    category VARCHAR(32),
    series1 FLOAT,
    series2 FLOAT,
    series3 FLOAT
);
```

加入样本数据：

```
INSERT INTO categorydata1 VALUES ('London', 54.3, 32.1, 53.4);
INSERT INTO categorydata1 VALUES ('New York', 43.4, 54.3, 75.2);
INSERT INTO categorydata1 VALUES ('Paris', 17.9, 34.8, 37.1);
```

15.4.5 创建XY图表数据

创建种类图数据库表：

```
CREATE TABLE xydata1 (
    date DATE,
    series1 FLOAT,
    series2 FLOAT,
    series3 FLOAT
);
```

加入样本数据：

```
INSERT INTO xydata1 VALUES ('1-Aug-2002', 54.3, 32.1, 53.4);
INSERT INTO xydata1 VALUES ('2-Aug-2002', 43.4, 54.3, 75.2);
INSERT INTO xydata1 VALUES ('3-Aug-2002', 39.6, 55.9, 37.1);
INSERT INTO xydata1 VALUES ('4-Aug-2002', 35.4, 55.2, 27.5);
INSERT INTO xydata1 VALUES ('5-Aug-2002', 33.9, 49.8, 22.3);
INSERT INTO xydata1 VALUES ('6-Aug-2002', 35.2, 48.4, 17.7);
INSERT INTO xydata1 VALUES ('7-Aug-2002', 38.9, 49.7, 15.3);
INSERT INTO xydata1 VALUES ('8-Aug-2002', 36.3, 44.4, 12.1);
INSERT INTO xydata1 VALUES ('9-Aug-2002', 31.0, 46.3, 11.0);
```

15.4.6 设置权限

最后一步是给样本数据授一读的权限给新用户jfreechart：

```
GRANT SELECT ON piedata1 TO jfreechart;
GRANT SELECT ON categorydata1 TO jfreechart;
GRANT SELECT ON xydata1 TO jfreechart;
```

15.5 JDBC驱动

为了使用JDBC访问样本数据，我们需要获得数据库的JDBC驱动。对于PostgreSQL，可以从下面的连接下载：

http://jdbc.postgresql.org

为了使用这个驱动，确保这个驱动jar文件加到classpath中。

15.6 应用演示

15.6.1 JDBC饼图演示

JDBC饼图演示实例将使用饼图数据表的数据产生饼图。该数据是由我们配置的数据库中获得的。

读数据的代码在方法readData()中：

```
private PieDataset readData() {
    JDBCPieDataset data = null;
    String url = "jdbc:postgresql://nomad/jfreechartdb";
    Connection con;
    try {
        Class.forName("org.postgresql.Driver");
    } catch (ClassNotFoundException e) {
        System.err.print("ClassNotFoundException: ");
        System.err.println(e.getMessage());
    }
    try {
        con = DriverManager.getConnection(url, "jfreechart", "passw
        data = new JDBCPieDataset(con);
        String sql = "SELECT * FROM PIEDATA1;";
        data.executeQuery(sql);
        con.close();
    } catch (SQLException e) {
        System.err.print("SQLException: ");
        System.err.println(e.getMessage());
    } catch (Exception e) {
        System.err.print("Exception: ");
        System.err.println(e.getMessage());
    }
    return data;
}
```

在代码中需要注意的事项：

- url是连接数据库的链接字符串。
- 返回的查询数据使用了JDBCPieDataset类对象进行了封装。详细内容见文档。

15.6.2 JDBC种类图演示

JDBC种类图应用使用种类数据产生了一个条形直方图。代码类似于JDBC饼图代码。但我们需要使用JDBCCategoryDataset类类封装格式化数据。

15.6.3 JDBC XY图演示

JDBC XY图应用使用XY数据产生了一个时序图。代码类似于JDBC饼图代码。但我们需要使用JDBCXYDataset类类封装格式化数据。

16 导出图表为PDF格式

16.1 简介

在本章节中，我们讲述如何将一个JFreeChart生成图表转换成PDF格式文件。主要使用的工具是IText。随着讲述，在后面的章节中用一个简单的实例说明创建PDF文件的过程，该PDF文件包含了一个简单的图表。生成的文件，可以使用Acrobat阅读器来阅读，也可以使用支持PDF文件阅读的阅读器来阅读。

16.2 什么是Acrobat PDF

Acrobat PDF是一款非常流行的电子文档阅读器。可实现不同的硬件平台和软件应用程序之间的信息共享,不受软件版本的不同和安装的字体的影响。PDF可以使用Adobe 提供的一个免费工具Acrodbat Reader来创建。Acrodbat Reader在终端用户各种平台上是有效的，包括GNU/Linux,Windows,Unix,Machintosh等。

如果你的系统上没有安装Acrodbat Reader，可以到下面链接去下载：

http://www.adobe.com/products/acrobat/readstep.html

16.3 IText

iText 是一个能够快速产生PDF文件的java类库。iText的主页下载地址是：

http://www.lowagie.com/iText

截止写本文的时间，最新的版本是2.0.6

16.4 Graphics2D

JFreeChart使用iText工具是非常容易的事情，因为iText提供了Graphics2D的实现。在我们说明实例应用之前，我们先回顾一下Graphics2D的类。

Java.awt. Graphics2D类，标准java2D API的一部分。定义了在二维空间中大量画文本和图形的方法。Graphics2D部分子类处理全部的转化细节，从输出（文本和图形）到具体设置的映射转化。

JFreeChart画图表时，仅仅使用Graphics2D定义的方法。这就意味着JFreeChart可以将图表输出到Graphics2D子类支持的任何设备。

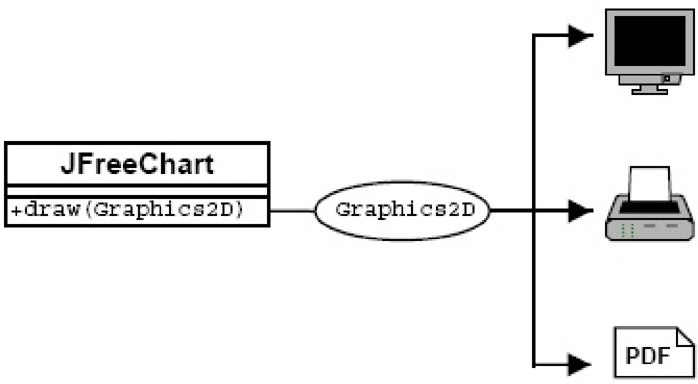


图16.2 JFreeChart画图的方法

iText工具融入了PdfGraphics2D的一个类，这就意味着iText使用Graphics2D类定义的方法产生PDF内容。并且正如你在后面的章节中看到的，在PDF格式中产生图表会变的非常的容易。

16.5 开始导出

为了完成和演示应用实例，我们需要下面的jar文件：

文件	描述
jfreechart-1.0.6.jar	JFreeChart类库
jcommon-1.0.9.jar	Jcommon类库
itext-2.0.6.jar	Itext类库

首先JFreeChart包括两个jar文件，其次iText需要一个jar文件。

16.6 实例应用

首先，需要创建一个图表，我们创建一个时序图，代码如下：

```
XYDataset dataset = createDataset();
JFreeChart chart = ChartFactory.createTimeSeriesChart(
    "Legal & General Unit Trust Prices",
    "Date",
    "Price Per Unit",
    dataset,
    true,
    true,
    false
);
```

这里没有任何的特殊代码——事实上，我们可以使用创建JFreeChart的其他对象替代上面的代码。

下一步，我们将在一个PDF文件中保存一个图表的副本：

```
File fileName = new File(System.getProperty("user.home") + "/jfreec
saveChartAsPDF(fileName, chart, 400, 300, new DefaultFontMapper());
```

下面有一些需要注意的问题：

首先，PDF文件名称是硬编码完成的，不能修改。主要是在演示中，减少代码量。在实际应用中，我们需要提供一些让用户指定文件名称与路径的方式，比如弹出一个文件选择对话框。

其次，saveChartAsPDF()方法还未实现。为了创建这个方法，我们先创建另一个更通用的writeChartAsPDF().方法。该方法执行saveChartAsPDF()方法需要的全部工作。但该方法的输入参数是一个文件输出流而不是一个文件，代码如下：

```
public static void writeChartAsPDF(OutputStream out, JFreeChart cha
int width, int height, FontMapper mapper) throws IOException {
    Rectangle pagesize = new Rectangle(width, height);
    Document document = new Document(pagesize, 50, 50, 50, 50);
    try {
        PdfWriter writer = PdfWriter.getInstance(document, out);
        document.addAuthor("JFreeChart");
        document.addSubject("Demonstration");
        document.open();
        PdfContentByte cb = writer.getDirectContent();
        PdfTemplate tp = cb.createTemplate(width, height);
        Graphics2D g2 = tp.createGraphics(width, height, mapper);
        Rectangle2D r2D = new Rectangle2D.Double(0, 0, width, heigh
        chart.draw(g2, r2D);
        g2.dispose();
        cb.addTemplate(tp, 0, 0);
    } catch (DocumentException de) {
        System.err.println(de.getMessage());
    }
    document.close();
}
```

在上面代码的方法里面，我们看到一些创建和代码iText文档的代码，从文档中获得了一个Graphics2D实例，使用Graphics2D对象画出这个图表，并关闭了这个文档。

同时我们也注意到方法的一个参数是FontMapper对象。iText使用FontMapper接口将java字体对象映射成基本的字体对象。DefaultFontMapper类预先默认映射为java本地化字体。如果你希望用这些字体，使用DefaultFontMapper构建缺省的对象即可，如果你相使用其他的字体（例如，支持一个特殊的字符集），那么我们需要做一些额外的工作。本章后面将有介绍。

在writeChartAsPDF()方法的实现里面，我们创建了一个自定义页面尺寸大小（匹配字符的需要尺寸）的PDF文档。我们提前设置了改变了字符的尺寸、位置并且在PDF文档中画出多个字符，以适应不同的页面尺寸。

现在我们将使用saveChartAsPDF()方法很容易的实现了将一个PDF数据发送到一个数据流上。建化了创建文件输出流的过程，并且将该对象传给了writeChartAsPDF()方法。代码如下：

```
public static void saveChartAsPDF(File file, JFreeChart chart, int
    int height, FontMapper mapper) throws IOException {
    OutputStream out = new BufferedOutputStream(new FileOutputStrea
    writeChartAsPDF(out, chart, width, height, mapper);
    out.close();
}
```

上面的每一步代码都是必须的。上面的代码组合成全部的代码如下（整个工程的代码都在这里，以便我们可以看到所有的声明和内容）：

```
package demo;
import java.awt.Graphics2D;
import java.awt.geom.Rectangle2D;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.DateAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.time.Month;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.xy.XYDataset;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Rectangle;
import com.lowagie.text.pdf.DefaultFontMapper;
import com.lowagie.text.pdf.FontMapper;
import com.lowagie.text.pdf.PdfContentByte;
import com.lowagie.text.pdf.PdfTemplate;
import com.lowagie.text.pdf.PdfWriter;
/**
* A simple demonstration showing how to write a chart to PDF format
* JFreeChart and iText.
* &lt;P&gt;
* You can download iText from http://www.lowagie.com/iText.
*/
public class PDFExportDemo1 {
    /**
    * Saves a chart to a PDF file.
    *
    * @param file
    * the file.
    * @param chart
    * the chart.
    * @param width
    * the chart width.
    * @param height
    * the chart height.
    */
    public static void saveChartAsPDF(File file, JFreeChart chart,
        int height, FontMapper mapper) throws IOException {
        OutputStream out = new BufferedOutputStream(new FileOutputS
        writeChartAsPDF(out, chart, width, height, mapper);
```

```
        out.close();
    }
    /**
    * Writes a chart to an output stream in PDF format.
    *
    * @param out
    * the output stream.
    * @param chart
    * the chart.
    * @param width
    * the chart width.
    * @param height
    * the chart height.
    *
    */
    public static void writeChartAsPDF(OutputStream out, JFreeChart
    int width, int height, FontMapper mapper) throws IOException {
        Rectangle pagesize = new Rectangle(width, height);
        Document document = new Document(pagesize, 50, 50, 50, 50);
        try {
            PdfWriter writer = PdfWriter.getInstance(document, out);
            document.addAuthor("JFreeChart");
            document.addSubject("Demonstration");
            document.open();
            PdfContentByte cb = writer.getDirectContent();
            PdfTemplate tp = cb.createTemplate(width, height);
            Graphics2D g2 = tp.createGraphics(width, height, mapper
            Rectangle2D r2D = new Rectangle2D.Double(0, 0, width, h
            chart.draw(g2, r2D);
            g2.dispose();
            cb.addTemplate(tp, 0, 0);
        } catch (DocumentException de) {
            System.err.println(de.getMessage());
        }
        document.close();
    }
    /**
    * Creates a dataset, consisting of two series of monthly data.
    *
    * @return the dataset.
    */
    public static XYDataset createDataset() {
        TimeSeries s1 = new TimeSeries("L&G European Index Trust",
        s1.add(new Month(2, 2001), 181.8);
        s1.add(new Month(3, 2001), 167.3);
        s1.add(new Month(4, 2001), 153.8);
        s1.add(new Month(5, 2001), 167.6);
        s1.add(new Month(6, 2001), 158.8);
        s1.add(new Month(7, 2001), 148.3);
        s1.add(new Month(8, 2001), 153.9);
        s1.add(new Month(9, 2001), 142.7);
        s1.add(new Month(10, 2001), 123.2);
        s1.add(new Month(11, 2001), 131.8);
```

```
        s1.add(new Month(12, 2001), 139.6);
        s1.add(new Month(1, 2002), 142.9);
        s1.add(new Month(2, 2002), 138.7);
        s1.add(new Month(3, 2002), 137.3);
        s1.add(new Month(4, 2002), 143.9);
        s1.add(new Month(5, 2002), 139.8);
        s1.add(new Month(6, 2002), 137.0);
        s1.add(new Month(7, 2002), 132.8);
        TimeSeries s2 = new TimeSeries("L&G UK Index Trust", Month.
        s2.add(new Month(2, 2001), 129.6);
        s2.add(new Month(3, 2001), 123.2);
        s2.add(new Month(4, 2001), 117.2);
        s2.add(new Month(5, 2001), 124.1);
        s2.add(new Month(6, 2001), 122.6);
        s2.add(new Month(7, 2001), 119.2);
        s2.add(new Month(8, 2001), 116.5);
        s2.add(new Month(9, 2001), 112.7);
        s2.add(new Month(10, 2001), 101.5);
        s2.add(new Month(11, 2001), 106.1);
        s2.add(new Month(12, 2001), 110.3);
        s2.add(new Month(1, 2002), 111.7);
        s2.add(new Month(2, 2002), 111.0);
        s2.add(new Month(3, 2002), 109.6);
        s2.add(new Month(4, 2002), 113.2);
        s2.add(new Month(5, 2002), 111.6);
        s2.add(new Month(6, 2002), 108.8);
        s2.add(new Month(7, 2002), 101.6);
        TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        return dataset;
    }
    public static void main(String[] args) {
        try {
            // create a chart...
            XYDataset dataset = createDataset();
            JFreeChart chart = ChartFactory.createTimeSeriesChart(
                "Legal & General Unit Trust Prices", "Date",
                "Price Per Unit", dataset, true, true, false);
            // some additional chart customisation here...
            XYPlot plot = chart.getXYPlot();
            XYLineAndShapeRenderer renderer = (XYLineAndShapeRende
                .getRenderer();
            renderer.setShapesVisible(true);
            DateAxis axis = (DateAxis) plot.getDomainAxis();
            axis.setDateFormatOverride(new SimpleDateFormat("MMM-yy
            // write the chart to a PDF file...
            File fileName = new File(System.getProperty("user.home"
                + "/jfreechart1.pdf");
            System.out.println(fileName.getPath());
            saveChartAsPDF(fileName, chart, 400, 300, new DefaultFo
        } catch (IOException e) {
            System.out.println(e.getMessage());
```

```
            }
        }
    }
```

在你完成和运行上面的应用之前，记得修改PDF文件的名称以满足我们的要求。同时前面16.5节提到的jar也必须在我们的classpath中。.

16.7 查看PDF 文件

在我们完成上面实例，运行实例，会产生一个PDF文档。我们可以使用一个PDF浏览器（比如AcrobatReader（或者其他支持的阅读器，Gnome PDF Viewer））查看该文件，显示的界面如下图16.3

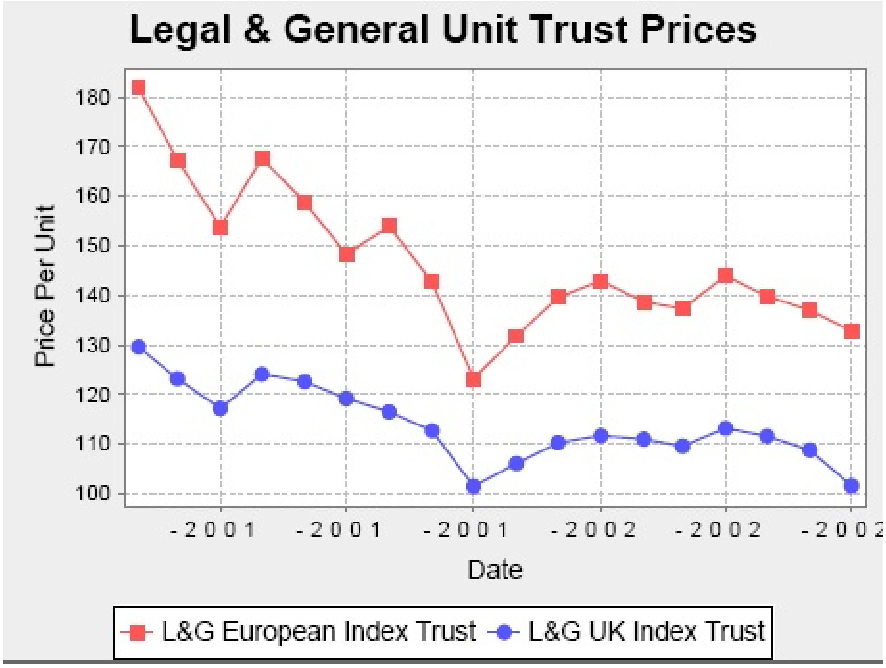


图16.3 JFreeChart使用iText生成的PDF文件图

大部分的PDF阅读器都提供了缩放技术，以允许我们更进一步浏览我们的图表。

16.8 Unicode字符问题

声明：由于本人对自字符集了解不深，因此翻译效果比较差，忘各大网友给予大力支持。

在我们关心我们所使用的字体字符集时，在JFreeChart和iText中使用Unicode字符集是没有任何问题的。在上面的例子中我们需要做一些修改来演示如何做到这些。

16.8.1 背景

Java使用同一的字符集译码成文本字符串。这种译码对每个字符使用16进制。这就意味着将有65，536个有效的不同字符集（在Unicode标准中定义了大约38，000个字符）。

我们可以在JFreeChart和iText中使用这些字符,但归于一条：那就是只要我们使用的字体，包括我们用来显示的文本或者不显示的，都必须定义这些字符。许多字体并不完全显示成Unicode字符集。下面的网站含有那些的确支持Unicode的字体的有用信息：

http://www.slovo.info/unifonts.htm

我们可以成功的提取使用tahoma.ttf字体。实际上，下面实例中我们将使用该字体，Tahoma字体并不是支持Unicode定义的每一个字符。因此，如果我们想使用一种特殊的字体时，就得必须选Unicode中一种相近的字体来代替。我们系统上都安装了字体Unicode MS (arialuni.ttf)——该字体完全支持Unicode字符集，尽管这种字体的定义的文件特别大（大约24M）

16.8.2 字体、iText和Java

iText依照PDF规格来处理字体，这就对使用PDF文件嵌入的字体来处理文件带来了非常大的方便性，同时也需要自由读取定义文件的字体。

而java在字体类中汲取了部分字体格式的大部分细节内容。

在iText中，为支持Graphics2D的实现画图功能，实现从Java字体对象到BaseFont对象的映射字体对象是非常有必要的。这就是FontMapper接口所扮演的角色。

如果我们使用缺省的构建器构建了一个新的DefaultFontMapper实例，那么总会带有Java规格定义的本地字体映射。但是如果我们想使用其他一些字体——并且我们必须使用Unicode之外的字符——那么我们需要将其他的字符映射加入到DefaultFontMapper对象中。

16.8.3 映射第三方的字体

这里我们决定使用Tahoma字体来显示标题。该字体的定义文件(tahoma.ttf)在我们系统下面的目录下可以找到。

```
/opt/sun-jdk-1.4.2.08/jre/lib/fonts
```

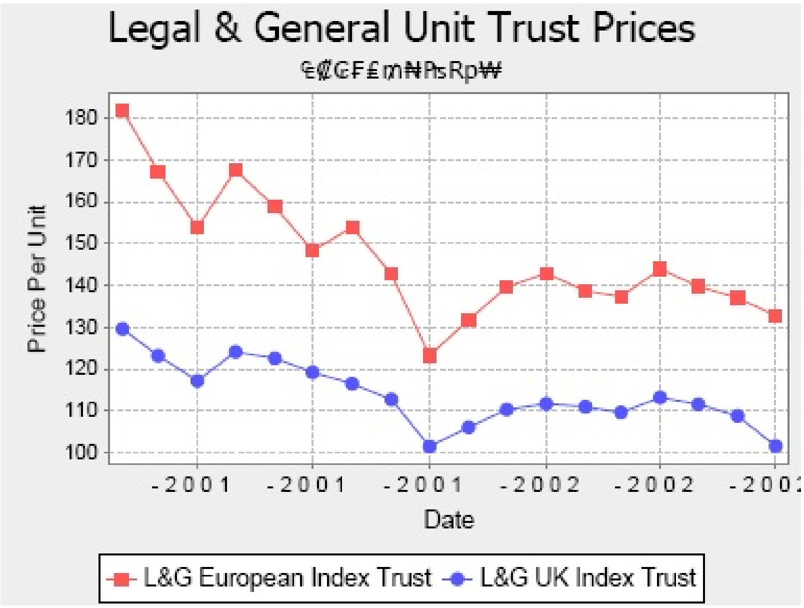
现在我们使用代码说明，使用iText创建FontMapper对象来使用Tahoma字体：

```
//设置字体
DefaultFontMapper mapper = new DefaultFontMapper();
mapper.insertDirectory("D:\\jre1.5.0_10\\lib\\fonts");
DefaultFontMapper.BaseFontParameters pp =
    mapper.getBaseFontParameters("Tahoma");
if (pp!=null) {
    pp.encoding = BaseFont.IDENTITY_H;
}
```

现在我们可以修改创建图表的代码，将图表的标题使用该字体：

```
TextTitle textTitle = chart.getTitle();
textTitle.setFont(new Font("Tahoma", Font.PLAIN, 20));
String text = "\u278A\u20A0\u20A1\u20A2\u20A3\u20A4\u20A5\u20A6\u20
// String text = "hi";
Font font = new Font("Tahoma", Font.PLAIN, 12);
TextTitle subtitle = new TextTitle(text, font);
chart.addSubtitle(subtitle);
```

副标题的输出如下图16.2所示。实例已经嵌入到PDF文件中。因此本文演示的该小程序很好的展示了这种类型的输出，给出了详细的步骤指南，便于我们正确使用。



如图16.2 一个Unicode副标题的图表

17 导出图表为SVG格式

17.1 简介

在本章里，我们介绍了一个简单实例，实例演示使用JFreeChart和Batik工具（SVG开源的类库）如何将图表导出为SVG格式。

17.2 背景

17.2.1 什么是SVG?

SVG(可放缩的矢量图形) 是W3C(World Wide Web ConSor—tium国际互联网标准组织)在2000年8月制定的一种基于XML格式的新的二维矢量图形格式，也是规范中的网络矢量图形标准。

17.2.2 Batik

Batik是一个java开源的工具包，允许我们产生SVG内容。可以从下面的链接获得有效的Batik：

http://xml.apache.org/batik

在写本文之前，Batik最新的版本是1.7

17.3 实例代码

17.3.1 JFreeChart 和Batik

JFreeChart和Batik兼容性非常好，因为：

- 因为JFreeChart画的所有图表的输出都是使用的Java的Graphics2D；
- Batik具体实现了Graphics2D产生SVG输出的功能（SVGGraphics2D）。

在本章节，使用一个简单的实例说明使用JFreeChart和Batik实现SVG的输出。关于该实例的详细技术详见下面链接：

http://xml.apache.org/batik/svggen.html

17.3.2 开始

首先，我们需要下载Batik并依照网站的指导进行安装。

确保下章节的例子能够正常运行，需要将下面的jar包加到我们的classpath：

文件	描述
jcommon-1.0.9.jar	JFreeChart的通用类包。
jfreechart-1.0.6.jar	JFreeChart的类包
batik-awt-util.jar	Batik实时运行文件
batik-dom.jar	Batik实时运行文件
batik-svggen.jar	Batik实时运行文件
batik-util.jar	Batik实时运行文件

17.3.3 实例应用

在我们的开发环境中创建一个工程，并且将上节列出的jar包添加到工程路径上，并输入下面代码：

```
package demo;
import java.awt.geom.Rectangle2D;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.Writer;
import org.apache.batik.dom.GenericDOMImplementation;
import org.apache.batik.svggen.SVGGraphics2D;
```

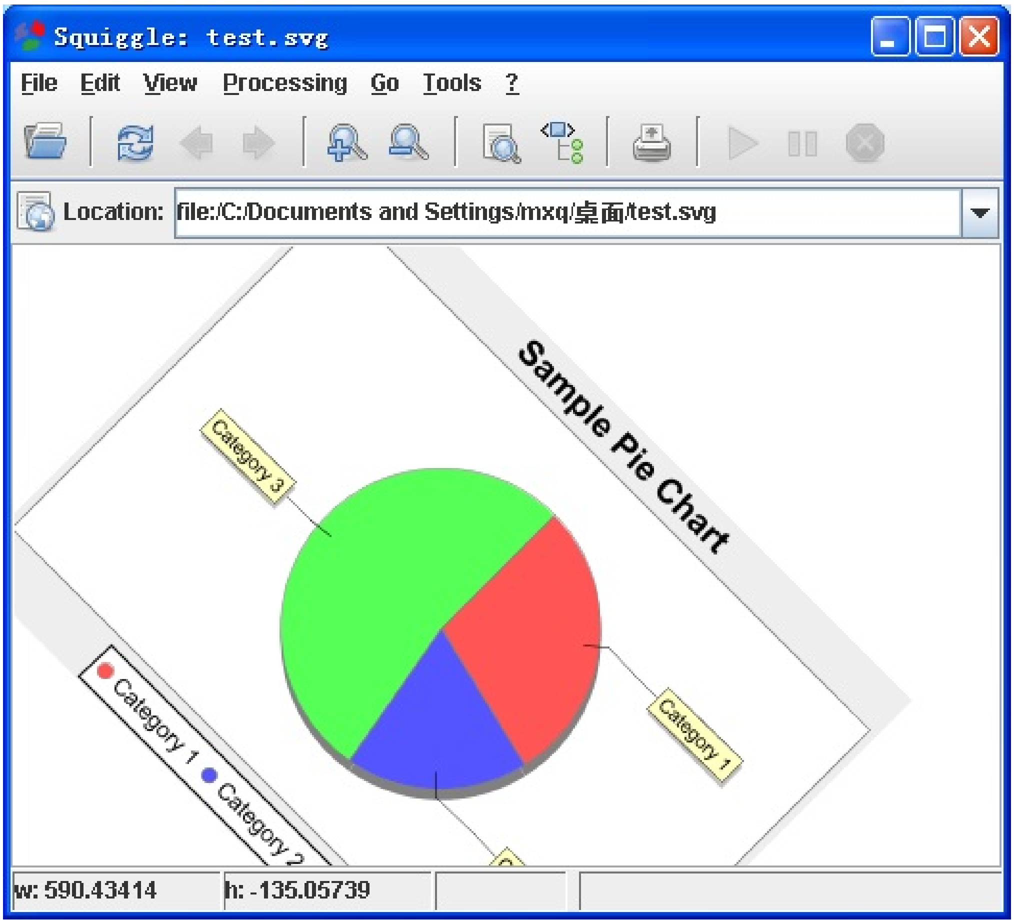
```
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.data.general.DefaultPieDataset;
import org.w3c.dom.DOMImplementation;
import org.w3c.dom.Document;
/**
* A demonstration showing the export of a chart to SVG format.
*/
    public class SVGExportDemo {
    /**
    * Starting point for the demo.
    *
    * @param args
    * ignored.
    */
    public static void main(String[] args) throws IOException {
        // create a dataset...
        DefaultPieDataset data = new DefaultPieDataset();
        data.setValue("Category 1", new Double(43.2));
        data.setValue("Category 2", new Double(27.9));
        data.setValue("Category 3", new Double(79.5));
        // create a chart
        JFreeChart chart = ChartFactory.createPieChart("Sample Pie
            data, true, false, false);
        // THE FOLLOWING CODE BASED ON THE EXAMPLE IN THE BATIK DOC
        // Get a DOMImplementation
        DOMImplementation domImpl = GenericDOMImplementation
            .getDOMImplementation();
        // Create an instance of org.w3c.dom.Document
        Document document = domImpl.createDocument(null, "svg", nul
        // Create an instance of the SVG Generator
        SVGGraphics2D svgGenerator = new SVGGraphics2D(document);
        // set the precision to avoid a null pointer exception in B
        svgGenerator.getGeneratorContext().setPrecision(6);
        // Ask the chart to render into the SVG Graphics2D implemen
        chart.draw(svgGenerator, new Rectangle2D.Double(0, 0, 400,
        // Finally, stream out SVG to a file using UTF-8 character
        // byte encoding
        boolean useCSS = true;
        Writer out = new OutputStreamWriter(new FileOutputStream(ne
            "test.svg")), "UTF-8");
        svgGenerator.stream(out, useCSS);
    }
}
```

17.3.4 浏览SVG图

Batik类库内包含了一个”Squiggle”的小应用，我们可以使用该工具浏览SVG文件。我们可以使用下面命令打开：

```
java -jar batik-squiggle.jar
```

下图截屏显示了上述代码创建的一个饼图的界面。使用应用浏览器工具，将图表在浏览器中进行了45度旋转。



如图17.1 SVG截图

18 Applet

18.1 简介

局限于一些条件，在Applet中使用JFreeChart还是比较容易的。本章节对Applet进行了整体的介绍，并举例说明了工作过程。这样对我们开始使用Applet提供极大帮助。

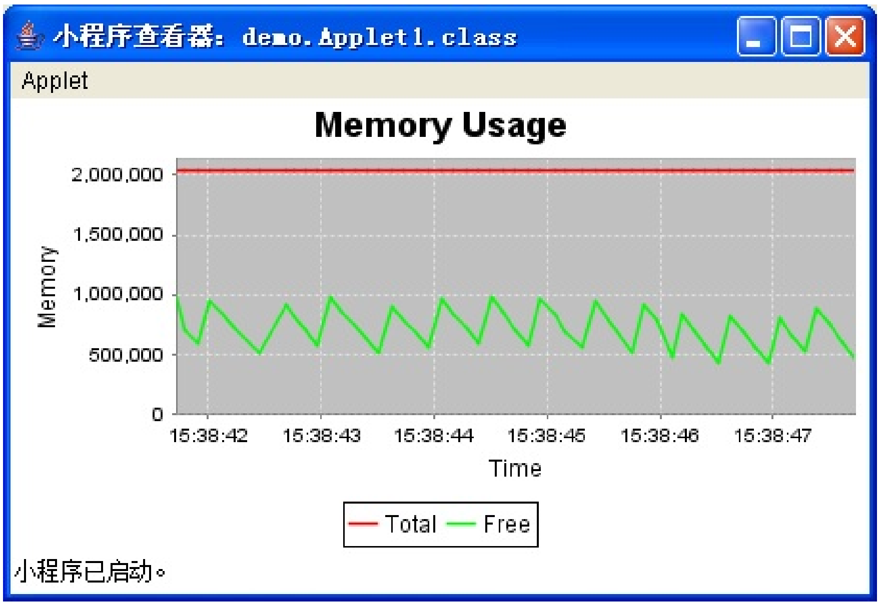


图18.1 JFreeChart在Applet上的应用

图18.1显示了一个使用JFreeChart的Applet简单应用。该applet可以通过下面链接看效果。

http://www.object-refinery.com/jfreechart/applet.html

后面的章节有全部的代码。

18.2 问题

在开发applet时，考虑的主要问题（与JFreeChart无关）是：

- 浏览器器支持
- 安全约束
- 字节码文件大小

在我们用提供的有效资源写applets时，确保我们对上面问题有所了解

18.2.1 浏览器支持

绝大部分的web浏览器均对最新版本的JDK1.5提供支持，因此使用JFreeChart运行applets也是绝对没有任何问题的（JFreeChart可以运行在JDK1.3.1版本或以上版本）。尽管如此，很大一部分用户还是使用一个浏览器的——微软的IE浏览器——该浏览器仅仅支持JDK1.1版本，并且现在已经过期。这里有一个问题是，使用JFreeChart的applet应用在微软的IE上是不能默认运行的。这就必须下载一个Java的插件，但是这样会造成很多不必要的麻烦和困难，最终的问题就是那些开发者选择写applets开发的问题，这导致开发者放弃开发applets，而选择Java Servlets（见下一章节）。

18.2.2 安全

Applets设计时，是符合java安全规范的。当一个applet运行在我们的web浏览器上时，在操作上是受到很大的约束的。例如，一个applet典型的是不能读写本地文件系统的。关于Java安全机制的描述已经超出了本章的范围，但是我们必须意识到JFreeChart的一些功能在applets上是不能运行的（例如将图表保存成PNG格式的文件），主要受java缺省的安全规则约束。如果我们想使用这些功能，那么我们需要认真学习一下java的安全机制的更多细节。

18.2.3 代码大小

最后一个文件就是我们applet运行时需要的代码量问题。在我们运行一个applet之前，代码被下载到本地客户端。显然对用户来说是有带宽限制的，代码量的大小成了关键问题。JFreeChart代码的jar文件大约是1M左右，对JFreeChart支持的图表来说，不算很大，但对使用modem拨号上网的用户来说，的确不是很理想的。同时我们需要将JCommon的jar包（大约290KB）加到我们的applet上。考虑到这些问题，我们将对JFreeChart进行重新打包，仅仅将applet需要的类文件包含进来，从而优化代码结构。

18.3 实例应用

正如在简介中所提及的，使用JFreeChart的applet可以在下面链接中看到：

http://www.object-refinery.com/jfreechart/applet.html

运行applet应用时，需要两个方面的支持。一是代码方式创建applet，二是HTML文件用来调用applet。

18.3.1 HTML

因为applet需要引入额外的jar文件，所以在HTML中使用applet是显得非常重要。HTML applet标签如下：

```
<APPLET ARCHIVE="jfreechart-1.0.6-applet-demo.jar,
jfreechart-1.0.6.jar,jcommon-1.0.9.jar"
CODE="demo.applet.Applet1" width=640 height=260
ALT="You should see an applet, not this text.">
</APPLET>
```

注意这里有三个jar需要引入，第一个包含了applet类，另外两个jar文件是JFreeChart和JCommon类库。我们需要在HTML文件中将applet标签引入。

18.3.2 源代码

实例applet的源代码见下（代码中我们使用了很少的applet特殊的代码，仅仅扩展了JApplet）：

```
package demo;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JApplet;
import javax.swing.Timer;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.DateAxis;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYItemRenderer;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.time.Millisecond;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
/**
```

```
* A simple applet demo.
*/
public class Applet1 extends JApplet {
    /** Time series for total memory used. */
    private TimeSeries total;
    /** Time series for free memory. */
    private TimeSeries free;
    /**
    * Creates a new instance.
    */
    public Applet1() {
        // create two series that automatically discard data more t
        // 30 seconds old...
        this.total = new TimeSeries("Total", Millisecond.class);
        this.total.setMaximumItemAge(30000);
        this.free = new TimeSeries("Free", Millisecond.class);
        this.free.setMaximumItemAge(30000);
        TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(total);
        dataset.addSeries(free);
        DateAxis domain = new DateAxis("Time");
        NumberAxis range = new NumberAxis("Memory");
        XYItemRenderer renderer = new XYLineAndShapeRenderer(true,
        XYPlot plot = new XYPlot(dataset, domain, range, renderer);
        plot.setBackgroundPaint(Color.lightGray);
        plot.setDomainGridlinePaint(Color.white);
        plot.setRangeGridlinePaint(Color.white);
        renderer.setSeriesPaint(0, Color.red);
        renderer.setSeriesPaint(1, Color.green);
        renderer.setSeriesStroke(0, new BasicStroke(1.5f));
        renderer.setSeriesStroke(1, new BasicStroke(1.5f));
        domain.setAutoRange(true);
        domain.setLowerMargin(0.0);
        domain.setUpperMargin(0.0);
        domain.setTickLabelsVisible(true);
        range.setStandardTickUnits(NumberAxis.createIntegerTickUnit
        JFreeChart chart = new JFreeChart("Memory Usage",
        JFreeChart.DEFAULT_TITLE_FONT, plot, true);
        chart.setBackgroundPaint(Color.white);
        ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setPopupMenu(null);
        getContentPane().add(chartPanel);
        new Applet1.DataGenerator().start();
    }
    /**
    * Adds an observation to the ’total memory’ time series.
    *
    * @param y
    * the total memory used.
    */
    private void addTotalObservation(double y) {
        total.add(new Millisecond(), y);
    }
```

```
    /**
    * Adds an observation to the ’free memory’ time series.
    *
    * @param y
    * the free memory.
    */
    private void addFreeObservation(double y) {
        free.add(new Millisecond(), y);
    }
    /**
    * The data generator.
    */
    class DataGenerator extends Timer implements ActionListener {
        /**
        * Constructor.
        */
        DataGenerator() {
            super(100, null);
            addActionListener(this);
        }
        /**
        * Adds a new free/total memory reading to the dataset.
        *
        * @param event
        * the action event.
        */
        public void actionPerformed(ActionEvent event) {
            long f = Runtime.getRuntime().freeMemory();
            long t = Runtime.getRuntime().totalMemory();
            addTotalObservation(t);
            addFreeObservation(f);
        }
    }
}
```

19 Servlets

19.1 介绍

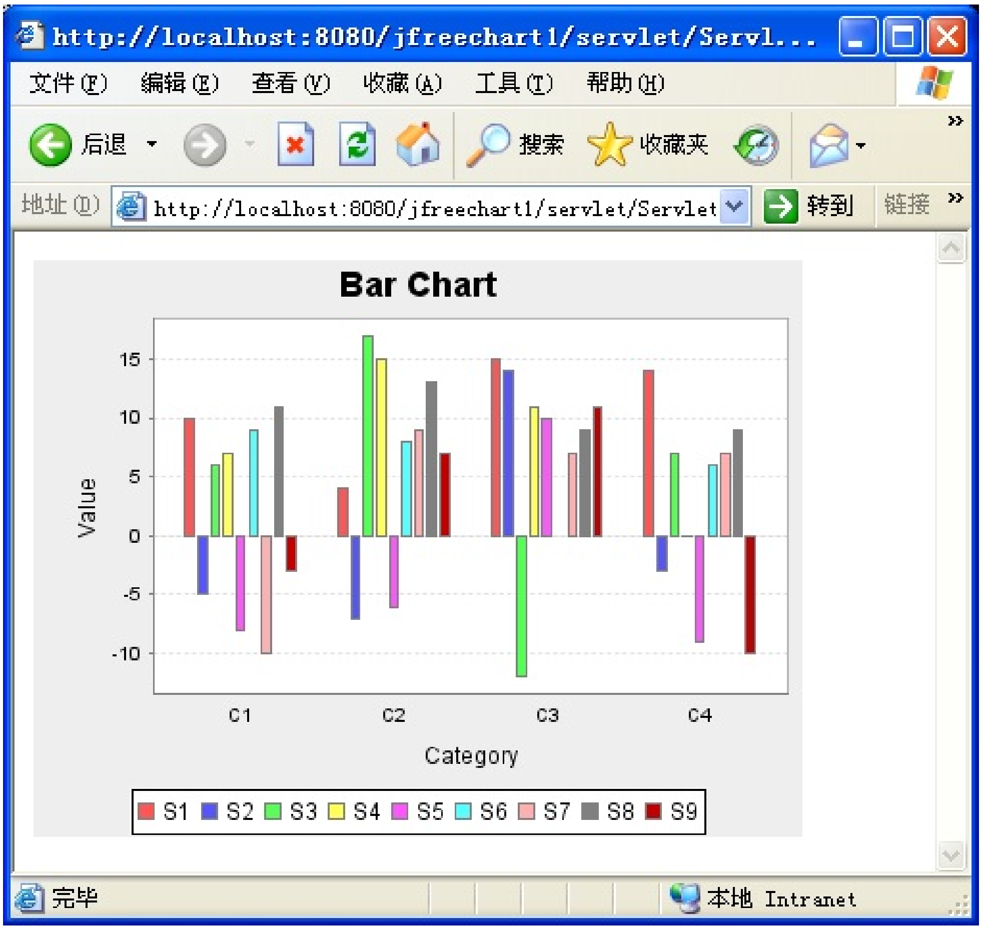
Java Servlet API是一套创建web应用非常流行成熟的技术。在servlet环境中使用JFreeChart是非常合适的。在本章节中，协助开发者在web应用中使用JFreeChart。本章所有的实例可以从下面链接中下载：

http://www.object-refinery.com/jfreechart/premium/index.html

下载的文件名为：jfreechart-1.0.6-demo.zip（该信息是收费的）.

19.2 编写一个简单的Servlet应用

ServletDemo1类实现了一个非常简单的servlet，该servlet返回了一个PNG图，PNG图是使用JFreeChart生成的直方条形图表。当运行该程序时，servlet在客户端仅仅显示一幅图片，而没有任何的HTML修饰，参见下图19.1。



如图19.1 浏览器中的servlet效果

我们以这种方式显示图片，是没有特殊意义，仅仅是为了：

- 很好演示servlets的请求－响应交互特征；
- 作为测试实例非常有用，我们会了解如何配置一个服务环境，如何让页面控件工作。

我们可以浏览后面更复杂的实例，显示使用HMTL表单如何请求不同的图表，并且将产生的图表的输出植入到HTML中。下面是基本servlet的代码。

```
package demo;
import java.io.IOException;
import java.io.OutputStream;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
/**
* A basic servlet that returns a PNG image file generated by JFreeC
* class is described in the JFreeChart Developer Guide in the "Serv
* chapter.
*/
public class ServletDemo1 extends HttpServlet {
    /**
    * Creates a new demo.
    */
    public ServletDemo1() {
    // nothing required
    }
    /**
    * Processes a GET request.
    *
    * @param request
    * the request.
    * @param response
    * the response.
    *
    * @throws ServletException
    * if there is a servlet related problem.
    * @throws IOException
    * if there is an I/O problem.
    */
    public void doGet(HttpServletRequest request, HttpServletRespon
    throws ServletException, IOException {
        OutputStream out = response.getOutputStream();
        try {
            DefaultCategoryDataset dataset = new DefaultCategoryDat
            dataset.addValue(10.0, "S1", "C1");
            dataset.addValue(4.0, "S1", "C2");
            dataset.addValue(15.0, "S1", "C3");
            dataset.addValue(14.0, "S1", "C4");
            dataset.addValue(-5.0, "S2", "C1");
            dataset.addValue(-7.0, "S2", "C2");
            dataset.addValue(14.0, "S2", "C3");
            dataset.addValue(-3.0, "S2", "C4");
            dataset.addValue(6.0, "S3", "C1");
            dataset.addValue(17.0, "S3", "C2");
            dataset.addValue(-12.0, "S3", "C3");
            dataset.addValue(7.0, "S3", "C4");
            dataset.addValue(7.0, "S4", "C1");
            dataset.addValue(15.0, "S4", "C2");
            dataset.addValue(11.0, "S4", "C3");
            dataset.addValue(0.0, "S4", "C4");
            dataset.addValue(-8.0, "S5", "C1");
            dataset.addValue(-6.0, "S5", "C2");
```

```
                dataset.addValue(10.0, "S5", "C3");
                dataset.addValue(-9.0, "S5", "C4");
                dataset.addValue(9.0, "S6", "C1");
                dataset.addValue(8.0, "S6", "C2");
                dataset.addValue(null, "S6", "C3");
                dataset.addValue(6.0, "S6", "C4");
                dataset.addValue(-10.0, "S7", "C1");
                dataset.addValue(9.0, "S7", "C2");
                dataset.addValue(7.0, "S7", "C3");
                dataset.addValue(7.0, "S7", "C4");
                dataset.addValue(11.0, "S8", "C1");
                dataset.addValue(13.0, "S8", "C2");
                dataset.addValue(9.0, "S8", "C3");
                dataset.addValue(9.0, "S8", "C4");
                dataset.addValue(-3.0, "S9", "C1");
                dataset.addValue(7.0, "S9", "C2");
                dataset.addValue(11.0, "S9", "C3");
                dataset.addValue(-10.0, "S9", "C4");
                JFreeChart chart = ChartFactory.createBarChart("Bar Cha
                    "Category", "Value", dataset, PlotOrientation.VERTI
                    true, true, false);
                response.setContentType("image/png");
                ChartUtilities.writeChartAsPNG(out, chart, 400, 300);
            } catch (Exception e) {
                System.err.println(e.toString());
            } finally {
                out.close();
            }
        }
    }
```

当一个客户端（通常是一个web浏览器）发出一个请求时，Servlet引擎调用DoGet()方法，以响应这个请求，servlet执行下面几步：

- 为客户端返回的输出获得一个输出流引用
- 创建一个图表；
- 响应的内容类型设置为image/png，这告诉客户端接受的数据类型是什么；
- 一个图表的PNG图表beii写进输出流；
- 输出流关闭。

19.3 编译实例Servlet

注意在javax.servlet.*包（包括子包）内的类，也就是实例使用的servlet，并不是J2SE的一部分。为了使用J2SE编译上面的代码需要另一个jar文件servlet.jar。我们使用了tomcat（是用java编写的一个开源servlet引擎）版本下面的这个servlet.jar文件。Tomcat可以在下面链接中获得：

http://tomcat.apache.org/

同时我们需要JFreeChat和JCommon两个类包文件来编译上面的代码。改变我们当前的工作目录，输入下面的命令（如果在Windows上，你需要将冒号“：“更改为分号“；“）

```
javac -classpath jfreechart-1.0.6.jar:lib/jcommon-1.0.9.jar:lib/se
source/demo/ServletDemo1.java
```

这样就生成了一个ServletDemo1.class文件，下一章内容描述了如何使用Tomcat部署这个servlet。

19.4 部署实例Servlet

Servlets部署在我们服务引擎的webapps的目录下面，在我们的例子中，使用的是Tomcat 5.5.20，将代码部署在：

```
D:\apache-tomcat-5.5.20\webapps\jfreechart1
```

在webapp目录下，创建一个目录jfreechart1来存放servlet演示实例，然后创建下面的结构目录。

```
.../jfreechart1/WEB-INF/web.xml
.../jfreechart1/WEB-INF/lib/jfreechart-1.0.6.jar
.../jfreechart1/WEB-INF/lib/jcommon-1.0.9.jar
.../jfreechart1/WEB-INF/classes/demo/ServletDemo1.class
```

我们需要创建web.xml文件——提供servlet的信息。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
    <servlet>
        <servlet-name>ServletDemo1</servlet-name>
        <servlet-class>demo.ServletDemo1</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ServletDemo1</servlet-name>
        <url-pattern>/servlet/ServletDemo1</url-pattern>
    </servlet-mapping>
</web-app>
```

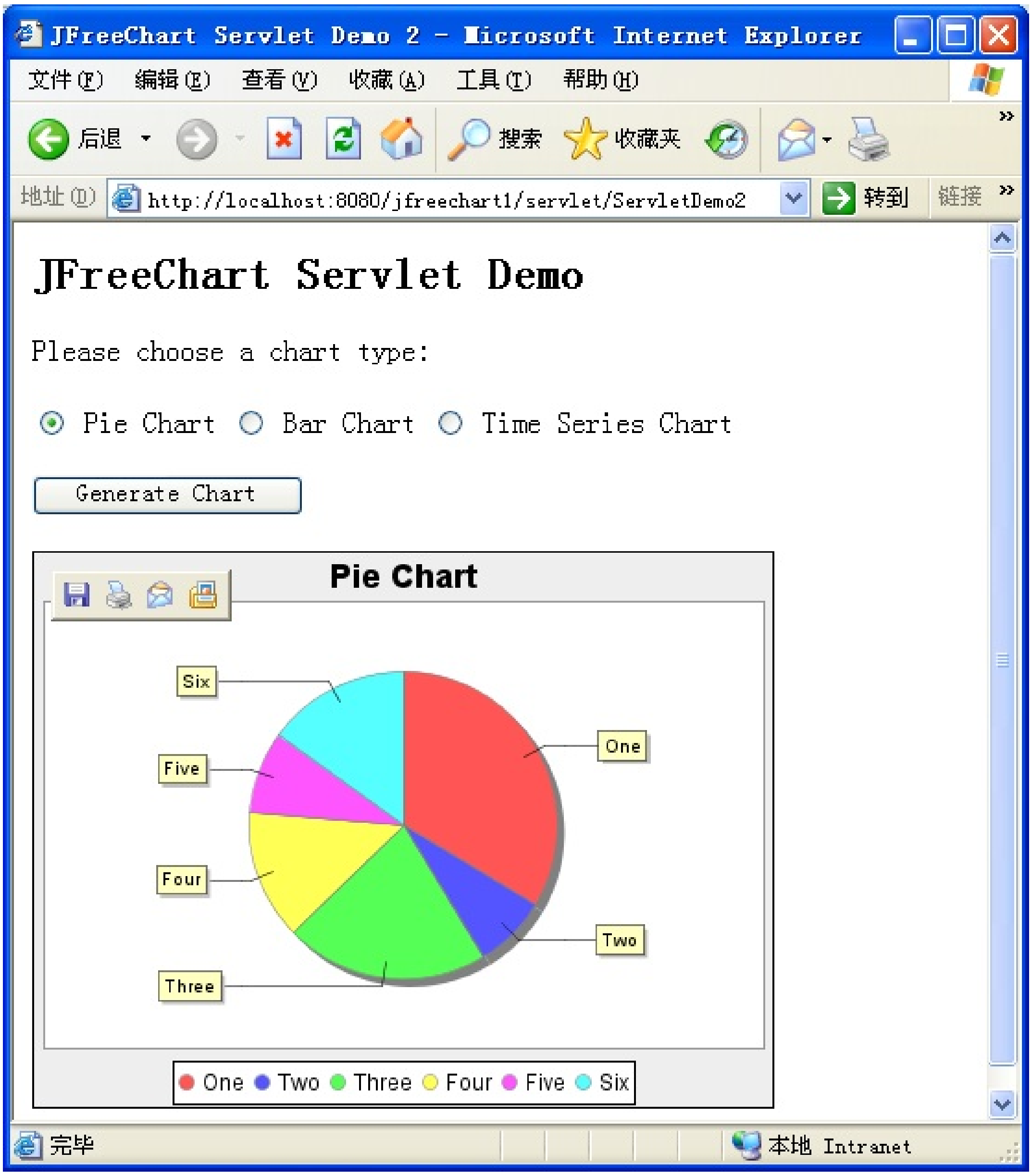
一旦上面的文件部署在servlet服务引擎上，然后启动我们的servlet服务引擎，在web浏览器中输入下面的地址：

http://localhost:8080/jfreechart1/servlet/ServletDemo1

如果不出现意外，我们就会在浏览器中看到如图19.1所示的界面。

19.5 在HMTL页面种嵌入图表

在HTML页面中嵌入servlet产生的图表图像也是可以的，下面实例ServletDemo2演示了这一特征。ServletDemo2实例处理一个HTML页面的请求，该HTML引用了另一个servlet（ServletDemo2ChartGenerator），引用的这个servlet返回了一个图表产生的PNG图像。最终的结果就是将图表嵌入到了一个HTML中，如图19.2所示：



如图19.2 浏览器中的ServletDemo2

全部代码如下：

```
package demo;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
* A basic servlet that generates an HTML page that displays a chart
* by JFreeChart.
* &lt;P&gt;
* This servlet uses another servlet (ServletDemo2ChartGenerator) to
* PNG image for the embedded chart.
* &lt;P&gt;
* This class is described in the JFreeChart Developer Guide.
*/
public class ServletDemo2 extends HttpServlet {
    /**
    *
    */
    private static final long serialVersionUID = 902404046769790985
    /**
    * Creates a new servlet demo.
    */
    public ServletDemo2() {
        // nothing required
    }
    /**
    * Processes a POST request.
    * &lt;P&gt;
    * The chart.html page contains a form for generating the first
    * after that the HTML returned by this servlet contains the sam
    * generating subsequent requests.
    *
    * @param request
    * the request.
    * @param response
    * the response.
    *
    * @throws ServletException
    * if there is a servlet related problem.
    * @throws IOException
    * if there is an I/O problem.
    */
    public void doPost(HttpServletRequest request, HttpServletRespo
        throws ServletException, IOException {
        PrintWriter out = new PrintWriter(response.getWriter());
        try {
            String param = request.getParameter("chart");
            response.setContentType("text/html");
            out.println("<HTML>");
            out.println("<HEAD>");
            out.println("<TITLE>JFreeChart Servlet Demo 2</TITLE>")
            out.println("</HEAD>");
            out.println("<BODY>");
            out.println("<H2>JFreeChart Servlet Demo</H2>");
            out.println("<P>");
            out.println("Please choose a chart type:");
```

```
            out.println("<FORM ACTION=\"ServletDemo2\" METHOD=POST>
            String pieChecked = (param.equals("pie") ? " CHECKED" :
            String barChecked = (param.equals("bar") ? " CHECKED" :
            String timeChecked = (param.equals("time") ? " CHECKED"
            out.println("<INPUT TYPE=\"radio\" NAME=\"chart\" VALUE
                + pieChecked + "> Pie Chart");
            out.println("<INPUT TYPE=\"radio\" NAME=\"chart\" VALUE
                + barChecked + "> Bar Chart");
            out.println("<INPUT TYPE=\"radio\" NAME=\"chart\" VALUE
                + timeChecked + "> Time Series Chart");
            out.println("<P>");
            out.println("<INPUT TYPE=\"submit\" VALUE=\"Generate Ch
            out.println("</FORM>");
            out.println("<P>");
            out.println("<IMG SRC=\"ServletDemo2ChartGenerator?type
                + "\" BORDER=1 WIDTH=400 HEIGHT=300/>");
            out.println("</BODY>");
            out.println("</HTML>");
            out.flush();
            out.close();
        } catch (Exception e) {
            System.err.println(e.toString());
        } finally {
            out.close();
        }
    }
}
```

注意该代码是如何从响应的参数获得一个引用的，而不是上面实例中的一个输出流。愿意是因为该servlet将返回HTML文本，与前章返回的二进制数据（一个PNG图片）不同。响应的类型设置成为text/html格式，因为servlet返回的是HTML文件。重要的一点就是HTML引用另一个servlet（ServletDemo2ChartGenerator）中的<IMG>标签，ServletDemo2ChartGenerator创建了必要的图表图片。HTML使用<FORM>元素来建立图表参数控制着实际图表的返回。

下面是ServletDemo2ChartGenerator的全部代码：

```
package demo;
import java.io.IOException;
import java.io.OutputStream;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.data.general.DefaultPieDataset;
```

```
import org.jfree.data.time.Day;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.xy.XYDataset;
import org.jfree.date.SerialDate;
/**
* A servlet that returns one of three charts as a PNG image file. T
* is referenced in the HTML generated by ServletDemo2.
* &lt;P&gt;
* Three different charts can be generated, controlled by the 'type'
* The possible values are 'pie', 'bar' and 'time' (for time series)
* &lt;P&gt;
* This class is described in the JFreeChart Developer Guide.
*/
public class ServletDemo2ChartGenerator extends HttpServlet {
    /**
    * Default constructor.
    */
    public ServletDemo2ChartGenerator() {
        // nothing required
    }
    /**
    * Process a GET request.
    *
    * @param request
    * the request.
    * @param response
    * the response.
    *
    * @throws ServletException
    * if there is a servlet related problem.
    * @throws IOException
    * if there is an I/O problem.
    */
    public void doGet(HttpServletRequest request, HttpServletRespor
    throws ServletException, IOException {
        OutputStream out = response.getOutputStream();
        try {
            String type = request.getParameter("type");
            JFreeChart chart = null;
            if (type.equals("pie")) {
                chart = createPieChart();
            } else if (type.equals("bar")) {
                chart = createBarChart();
            } else if (type.equals("time")) {
                chart = createTimeSeriesChart();
            }
            if (chart != null) {
                response.setContentType("image/png");
                ChartUtilities.writeChartAsPNG(out, chart, 400, 30(
            }
        } catch (Exception e) {
            System.err.println(e.toString());
```

```
        } finally {
            out.close();
        }
    }
    /**
    * Creates a sample pie chart.
    *
    * @return a pie chart.
    */
    private JFreeChart createPieChart() {
        // create a dataset...
        DefaultPieDataset data = new DefaultPieDataset();
        data.setValue("One", new Double(43.2));
        data.setValue("Two", new Double(10.0));
        data.setValue("Three", new Double(27.5));
        data.setValue("Four", new Double(17.5));
        data.setValue("Five", new Double(11.0));
        data.setValue("Six", new Double(19.4));
        JFreeChart chart = ChartFactory.createPieChart("Pie Chart",
            true, false);
        return chart;
    }
    /**
    * Creates a sample bar chart.
    *
    * @return a bar chart.
    */
    private JFreeChart createBarChart() {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset
        dataset.addValue(10.0, "S1", "C1");
        dataset.addValue(4.0, "S1", "C2");
        dataset.addValue(15.0, "S1", "C3");
        dataset.addValue(14.0, "S1", "C4");
        dataset.addValue(-5.0, "S2", "C1");
        dataset.addValue(-7.0, "S2", "C2");
        dataset.addValue(14.0, "S2", "C3");
        dataset.addValue(-3.0, "S2", "C4");
        dataset.addValue(6.0, "S3", "C1");
        dataset.addValue(17.0, "S3", "C2");
        dataset.addValue(-12.0, "S3", "C3");
        dataset.addValue(7.0, "S3", "C4");
        dataset.addValue(7.0, "S4", "C1");
        dataset.addValue(15.0, "S4", "C2");
        dataset.addValue(11.0, "S4", "C3");
        dataset.addValue(0.0, "S4", "C4");
        dataset.addValue(-8.0, "S5", "C1");
        dataset.addValue(-6.0, "S5", "C2");
        dataset.addValue(10.0, "S5", "C3");
        dataset.addValue(-9.0, "S5", "C4");
        dataset.addValue(9.0, "S6", "C1");
        dataset.addValue(8.0, "S6", "C2");
        dataset.addValue(null, "S6", "C3");
        dataset.addValue(6.0, "S6", "C4");
```

```
        dataset.addValue(-10.0, "S7", "C1");
        dataset.addValue(9.0, "S7", "C2");
        dataset.addValue(7.0, "S7", "C3");
        dataset.addValue(7.0, "S7", "C4");
        dataset.addValue(11.0, "S8", "C1");
        dataset.addValue(13.0, "S8", "C2");
        dataset.addValue(9.0, "S8", "C3");
        dataset.addValue(9.0, "S8", "C4");
        dataset.addValue(-3.0, "S9", "C1");
        dataset.addValue(7.0, "S9", "C2");
        dataset.addValue(11.0, "S9", "C3");
        dataset.addValue(-10.0, "S9", "C4");
        JFreeChart chart = ChartFactory.createBarChart3D("Bar Chart
            "Category", "Value", dataset, PlotOrientation.VERTICAL,
            true, false);
        return chart;
    }
    /**
    * Creates a sample time series chart.
    *
    * @return a time series chart.
    */
    private JFreeChart createTimeSeriesChart() {
        // here we just populate a series with random data...
        TimeSeries series = new TimeSeries("Random Data");
        Day current = new Day(1, SerialDate.JANUARY, 2001);
        for (int i = 0; i &lt; 100; i++) {
            series.add(current, Math.random() * 100);
            current = (Day) current.next();
        }
        XYDataset data = new TimeSeriesCollection(series);
        JFreeChart chart = ChartFactory.createTimeSeriesChart(
            "Time Series Chart", "Date", "Rate", data, true, true,
        return chart;
    }
}
```

下一章讲述servlet的支持文件， 与如何部署它们。

19.6 支持文件

Servlet为客户端产生典型的输出。大部分web应用之少包含一个HTML文件，用来进入应用的入口。本章演示的servlet，使用index.html页面，代码如下：

```
<HTML>
<HEADER>
    <TITLE>JFreeChart : Basic Servlet Demo</TITLE>
</HEADER>
<BODY>
    <H2>JFreeChart: Basic Servlet Demo</H2>
    <P>There are two sample servlets available:
    <ul>
        <li>a very basic servlet to generate a <a
        href="servlet/ServletDemo1">bar chart;</li>
        <li>another servlet that allow you to select one of <a
        href="chart.html">three sample charts. The selected chart i
        displayed in an HTML page.</li>
    </ul>
    </BODY>
</HTML>
```

该页面上有两个链接，一个是实例1（ServletDemo1），第二个链接是两一个HTML页面，chart.html。代码如下：

```
<HTML>
<HEADER>
    <TITLE>JFreeChart Servlet Demo 2</TITLE>
</HEADER>
<BODY>
    <H2>JFreeChart Servlet Demo</H2>
    <P>Please choose a chart type:
    <FORM ACTION="servlet/ServletDemo2" METHOD=POST>
        <INPUT TYPE="radio" NAME="chart" VALUE="pie" CHECKED> Pie C
        <INPUT TYPE="radio" NAME="chart" VALUE="bar"> Bar Chart
        <INPUT TYPE="radio" NAME="chart" VALUE="time"> Time Series
        <INPUT TYPE="submit" VALUE="Generate Chart">
    </FORM>
</BODY>
</HTML>
```

第二个HTML页面包含<FORM>元素用来为第二个serlvet指定一个参数。当servlet运行时，返回自己的HTML，THML包含一个<IMG>元素，该元素引用了ServletDemo2ChartGenerator 的servlet。

19.7 部署Servlets

完成上面实例代码的编译厚，需要将它们连同支持文档部署到servlet引擎上，以便于客户端能够正确访问。幸运的是，这些都非常容易做到。

首先是配置web.xml文件，该文件用来描述web应用部署。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
    <servlet>
        <servlet-name>ServletDemo1</servlet-name>
        <servlet-class>demo.ServletDemo1</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ServletDemo2</servlet-name>
        <servlet-class>demo.ServletDemo2</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ServletDemo2ChartGenerator</servlet-name>
        <servlet-class>demo.ServletDemo2ChartGenerator</servlet-cla
    </servlet>
    <servlet-mapping>
        <servlet-name>ServletDemo1</servlet-name>
        <url-pattern>/servlet/ServletDemo1</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ServletDemo2</servlet-name>
        <url-pattern>/servlet/ServletDemo2</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ServletDemo2ChartGenerator</servlet-name>
        <url-pattern>/servlet/ServletDemo2ChartGenerator</url-patte
    </servlet-mapping>
</web-app>
```

该文件通过名字列出了全部的sevlets，并且指定了具体类。实际的类被放置在servlet引擎的指定目录下面。

最后的步骤是将全部的文档拷贝到响应的servlet引擎的目录下面。我们使用的是servlet引擎是Tomcat。在Tomcat下的webapps目录下面创建一个jfreechart2的目录，将index.html和chart.html文件拷贝到下面的目录：

```
webapps/jfreechart2/index.html
webapps/jfreechart2/chart.html
```

接下来，在目录jfreechart2下建立一个子目录WEB-INFO，将web.xml文件拷贝到该目录下面。

```
webapps/jfreechart2/WEB-INF/web.xml
```

WEB-INFO目录下面创建子目录classes/demo将编译的类放在该目录下面。

```
webapps/jfreechart2/WEB-INF/classes/demo/ServletDemo1.class
webapps/jfreechart2/WEB-INF/classes/demo/ServletDemo2.class
webapps/jfreechart2/WEB-INF/classes/demo/ServletDemo2ChartGenerator
```

最后，将相关的jar拷贝到下面目录：

```
webapps/jfreechart2/WEB-INF/lib/jcommon-1.0.9.jar
webapps/jfreechart2/WEB-INF/lib/jfreechart-1.0.6.jar
```

现在启动我们的servlet引擎，在我们的浏览器中输入：

http://localhost:8080/jfreechart2/index.html

如果全部文件放置在适当位置，而不出现特殊意外的话，我们将会看到上面图19.2所示的界面。

20 JFreeChart相关技术

20.1 简介

本章节主要介绍了JFreeChart涉及的各种信息。

20.2 X11/Headless Java

如果我们在Unix/Linux上的服务环境使用JFreeChart，我们会遇到JFreeChart在没有X11的情况下不能运行。这与运行在AWT上的java代码是同一个问题。更多的信息见下面链接：

http://java.sun.com/products/java-media/2D/forDevelopers/java2dfaq.html#xvfb

同时在JFreeChart的论坛里面也有好多的信息，可以找到一些额外的思路：

http://www.jfree.org/phpBB2/viewtopic.php?t=1012

20.3 JSP

如果开发者在JSP中使用JFreeChart比较感兴趣，那么可以从下面网址找到更多信息：

http://cewolf.sourceforge.net/

20.4 加载图片

图像在Java中是用Image来描述的。我们可以使用开发包里面的createImage()方法来创建图像，但是我们需要意识到该方法加载图像时不是同步的——换句话说，方法返回的与图像加载是在不同的线程中。这样会产生这样一个问题就是在图像为完全加载完成时，我们就使用了该图像。

我们可以使用MediaTracker类来检查图像加载的进度。但万一在某个地方我们需要在使用方法之前，必须确保图片完全加载完毕。这时该怎么办呢？使用Swing的ImageIcon类可以解决我们这个问题，代码如下：

```
ImageIcon icon = new ImageIcon("/home/dgilbert/temp/daylight.png");
Image image = icon.getImage();
```

构造方法直到图片完全加载完成后才返回，因此在我们调用方法getImage()时，下图已经图像加载完毕。

21 包

21.1 概述

下面内容讲述了JFreeChart类的参考信息。

包名	说明
org.jfree.chart	主图表类
org.jfree.chart.annotations	注释图表的简单框架
org.jfree.chart.axis	轴类和相关接口
org.jfree.chart.editor	为图表提供的属性编辑器框架（不完善）
org.jfree.chart.encoders	写图象文件类
org.jfree.chart.entity	描述图表实体的类
org.jfree.chart.event	事件类
org.jfree.chart.imagemap	HTML图片映像工具类
org.jfree.chart.labels	图表标签和信息提示类
org.jfree.chart.needle	Needle classes for the compass plot
org.jfree.chart.plot	Plot类和接口
org.jfree.chart.renderer	Renderer的基本类包
org.jfree.chart.renderer.category	Plug-in renderers for use with the CategoryPlot class.
org.jfree.chart.renderer.xy	Plug-in renderers for use with the XYPlot class.
org.jfree.chart.servlet	Servlet utility classes.
org.jfree.chart.title	图表标题类
org.jfree.chart.urls	在图像映像区产生URLs的接口和类
org.jfree.chart.util	实用工具类
org.jfree.data	Dataset接口和类
org.jfree.data.category	CategoryDataset接口和相关类
org.jfree.data.contour	ContourDataset接口和相关类
org.jfree.data.function	Function2D接口和相关类
org.jfree.data.gantt	甘特图的dataset接口和类
org.jfree.data.general	通用的dataset类

org.jfree.data.io	通用的dataset的I/O类
org.jfree.data.jdbc	JDBC相关的dataset类
org.jfree.data.statistics	产生统计的相关类
org.jfree.data.time	基于时间的dataset接口和类
org.jfree.data.time.ohlc	展示高低开发图表dataset的类
org.jfree.data.xml	从xml文件读取dataset的类
org.jfree.data.xy	XYDataset接口和相关类

更多的信息可以查看javadoc产生的HTML文档。